Elementary Statistical Modeling for Applied Biostatistics

Copyright 2018 Jeffrey A. Walker

Draft: 2019-11-20

Contents

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 something like a look-up table or a dichotomous key 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" something or "shown" an effect if $p < 0.05$ but found "no effect" if $p > 0.05$.

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. But if we are investigating the effects of an increasingly acidified ocean on coral growth, $p = 0.002$ may be evidence of an effect of the experimental intervention, but, from everything we know about pH and cell biology, it would be absurd to conclude from any data that pH does not affect growth. To build useful models of how biological systems work, we want to know the magnitude of effects and our uncertainty in estimating these magnitudes. 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.

This book is an introduction to the estimation of effects of biological data, and

measures of the uncertainty of theses estimates, using a statistical modeling approach. As an introduction, the focus will be linear models and extensions of the linear models including linear mixed models and generalized linear models. Linear models are the engine behind many hypothesis tests but the emphasis in statistical modeling is estimation and uncertainty instead of test statistics and *p*-values. All linear models, and their generalizations, are variations of

$$y_i \sim N(\mu_i, \theta) \tag{1}$$
$$\mathrm{E}(Y|X) = \mu \tag{2}$$
$$\mu_i = f(\beta_0 + \beta_1 x_i) \tag{3}$$

Chapter 1 explains the meaning of this **model specification** but the point to make here is that because all linear models and their generalizations are variations of this specification, a modeling strategy of learning or doing statistics is more coherent than the NHST strategy using look-up tables or dichotomous keys of hypothesis tests. Generalizations of the basic linear model include 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, instead, *a path of the critical elements leading you to the doorway to the vast universe of each of these methods.*

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.

---

[1]Gegenrezer

0.1 Math

0.2 R and programming

Part I: R fundamentals

Chapter 1

Organization – R Projects and R Notebooks

1.1   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. 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. data.table – improves functionality of data frames
2. readxl – elegant importing from microsoft Excel spreadsheets
3. janitor – we use the function clean_names from this package
4. ggplot2 – we use this for plotting
5. ggpubr – we use this to make ggplots a bit easier
6. emmeans – we use this to compute modeled means and contrasts
7. nlme, lme4, lmerTest – these are packages for multilevel modeling
8. MASS – we will use glm.nb from this package

Again, once these are installed, you don't need to do this again. You simply need to use the `library()` function at the start of a markdown script.

1.2   Create an R Studio Project for this Class

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 should be named "BIO_413").
2. Within the project folder, create new folders named
   1. "notebooks" – this is where your R notebooks are stored
   2. "R" – this is where additional R scripts 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

1.3   R Notebooks

A typical statistical modeling project will consist of:

1. importing data from Excel or text (.csv or .txt) files
2. cleaning data
3. analysis
4. model checking
5. generating plots
6. generating tables
7. 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 your R Notebook. 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.

The goal with an R Studio Notebook is to explicitly link all this so that changes in earlier steps automatically flow into the later steps. So, at the end of a project, a researcher can choose "run all" from the menu and the data are read, cleaned, analyzed, ploted, tabled, and put into a report with the text.

This means that you have to think of the organization of the R code that your write in a Notebook. Your cannot simply append new code to the end of a script if something earlier (or above) is dependent on it. You need to go back up and insert the new code at some earlier (and meaningful) point.

For example, an R chunk generates 100 random normal values and then plots these with a histogram. This was the chunk that I wrote

```
x <- rnorm(n)
qplot(x)
```

When I ran the chunk, I got the error "Error in rnorm(n) : object n not found". I was using the function `rnorm()` to generate values but I hadn't assigned any value to `n` yet, so I got the error. To get this to work properly, I could have just typed `n <- 100` in the console and then re-run the script but I want it to work properly on a fresh run of the chunk (after quitting and re-opening R Studio) so I instead inserted `n <- 100` at the start of the chunk, like this:

```
n <- 100
x <- rnorm(n)
qplot(x)
```

1.3.1 Create an R Notebook 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 Notebook" from the pull-down menu.
2. Change the title of the notebook to "Notebook_01-organization"
3. Delete the default R Markdown text starting with "This is an [R Markdown]…"

Now write some text documenting which packages you installed.

1.3.2 Create a "load-packages" chunk

1. Click on the "Insert" menu on the right hand side of the script (R Markdown) pane and choose "R". This will insert an R code chunk into your R markdown document.
2. The first R chunk of a notebook should be a setup chunk. Name the chunk "setup"
3. load the libraries ggplot2 and data.table and click the chunk's run button (the green triangle to the right of the chunk)

```r
library(ggplot2)
library(data.table)
```

I added the chunk option "message=FALSE". Run your chunk with and without this as an option.

1.3.3 Create a "simple plot" chunk

4. Create a new chunk and label it "simple plot"
5. insert the following R script and then click the chunk's run button. Do you get a plot?

```r
n <- 100
x <- rnorm(n)
qplot(x)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```
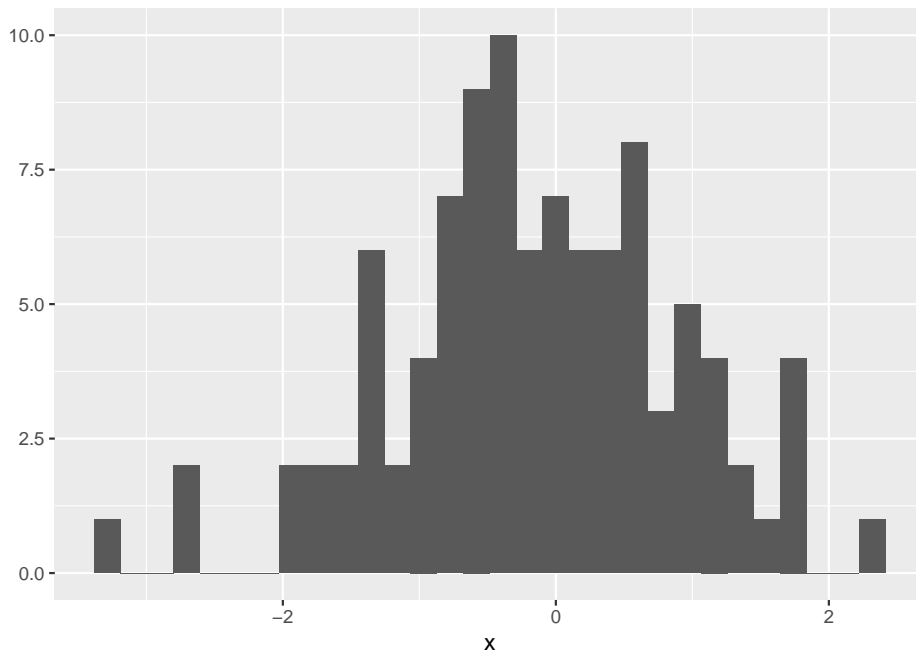

1.3.4 Create more R chunks and explore options and play with R code

Chapter 2

Data – Reading, Writing, and Wrangling

Importing data into R can be a struggle for new R users and, unfortunately, most online 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. Some examples:

(tl;dr – go to next section if you just want to import a file and don't want to understand this important background information!)

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_aroostook.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 notebook .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,

17

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 reproducility. 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 a notebook file is located within the notebooks 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 notebooks 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 imediately inside the project folder. This should work on any machine, and should work even if a project folder is moved. However, it does not work when writing a book with Bookdown a Bookdown project has multiple .Rmd files and the chunk names cannot be repeated across files. There can only be one chunk called "setup".

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_path <- "data" # path to data that are imported
file_name <- "my_data.xlsx"
file_path <- here(data_path, file_name) # paste together parts of the address
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 or to another machine. And no chunk named setup is required so this also works in every R markdown file when using Bookdown.

2.1 Create new notebook for this chapter

Create a new notebook and save it to the "notebooks" folder of your BIO_413 project. **Important:** In an R Studio project, the working directory is the project directory (the location of the .Rproj file) – this will be the working directory of the console. 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 an notebook code chunk is the folder containing the saved notebook file. Importantly, the working directory of the code chunk isn't "moved" to the folder containing the saved file *until the file is saved* so import/export scripts will no work properly until the file is saved!

Annotate your notebook with notes! Update it as you learn more! We will use data.table for importing text files in tab-delimited or comma-separated formats and the readxl package for importing excel files.

Start each notebook with a chunk named "setup" that loads packages and sets the address for reading and writing files (note that since this book is advocating

the use of here() to set the file path, naming the setup chunk "setup" is not imperative).

```r
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) # make data.frames data.tables
library(ggplot2) # ggplot environment
library(ggpubr) # publication ready plots
library(emmeans) # get estimated marginal means and CIs, used for plot

here <- here::here # plyr also defines a here function

# relative paths to project folders
data_path <- "data" # path to data that are imported
output_path <- "output" # path to data that are saved
image_path <- "images" # path to image folder
```

2.2   Importing data

Throughout this book, we will download data from the Dryad Digital Repository, which is a major resource for increasing reproducibility in science. My own view is that *all data* should be archived on some public server (exceptions include data that are proprietary or contain sensitive information – such as human health measures).

2.2.1   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//handle/10255/dryad.122315

Steps

1. Copy the title of the Dryad page, which is "Data from: Prenatal acoustic communication programs offspring for high post-hatching temperatures in a songbird"

2. Create a new folder within "data" and paste in the copied title as the folder name
3. Remove the colon from the name, so the folder name is "Data from Prenatal acoustic communication programs offspring for high post-hatching temperatures in a songbird"
4. 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 specialized software. 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.

```
data_folder <- "Data from Prenatal acoustic communication programs offspring for high post-hatchi
fn <- "allDatasetsMarietteBuchanan2016.xls"
file_path <- here(data_path, data_folder, fn)
#file_path <- paste(data_path, data_folder, fn, sep="/")
chick <- read_excel(file_path, sheet="nestlingMass") # read the excel file
chick <- data.table(chick) # convert to data.table
```

In this book, we will consistently uses this protocol for storing and retrieving downloaded files. The first three lines in the script above creates the directory path to the file. This path includes

1. data_path – the relative path into the folder "data" (relative to the location of the project file)
2. data_folder – the name of the folder within "data" containing the file
3. filename – the name of the file to read

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 `read_excel(file_path, sheet="nestlingMass")` reads the nestling-Mass sheet only. The next line:`chick <- data.table(chick)` converts the data.frame into a data.table.

2.2.1.1   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 "load-libraries" chunk and that you have run the chunk
2. Make sure you have assigned `data_path <- "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 is one level inside your "data" folder
5. Make sure you have the name of the "Data from ..." folder correct in your script. To do this, I do not type the name but instead go to the finder (on Mac OS) and highlight the "Data from ..." folder and copy the name – the whole name – and then paste in between the quote mark in the line `data_folder <- ""`
6. Make sure your data file is one level inside the correct "Data from" folder.
7. 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, R is *very literal* when it comes to: Spelling. Humans are very good at understanding misspelled words but the R language (or any computer language) is very literal. "Data from Quantifying the effects of" and "Data from Quantifying the efects of" are different values. Capitalization. R is **case sensitive** (some programming languages are not). "Data from Quantifying", "Data from quantifying", and "data from quantifying" are all different values. Spelling AND capitalization have to be perfect, not simply close. Humans are very good at understanding misspelled and OdDLy capitalized words but the R language (or any computer language) is very literal.

2.2.1.2   Peak at the imported data.table to check that the file was imported correctly and to learn about the contents.

Insert the following after your import script and run:

```r
View(chick) # check -- are there headers? are there the correct number of columns?
```

The line `View(chick)` script opens a new tab containing a spreadsheet-like display of the data frame `chick`. This is one way to check that the data were imported correctly, to examine the column labels to ensure they conform to best practices, and to simply get to know the contents of the imported file. From the view of chick, it may not be immediately apparent that the column names contain spaces. This can be seen more easily using

```r
colnames(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 `brood_id`. Some coders separate words with a period (`brood.id`). Others mash words together into a single word like this `broodid` but this should generally be avoided because the result can be hard to read. Finally, some coders use Caps to designate new words like this `BroodId`. This is easier to read than simple concatenation but the underscore is the easiest to read.

The `janitor` package has a nice 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.

```r
chick <- clean_names(chick)
```

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

2.2.1.3 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 excelllent review of best practices for organizing data in a spreadsheet.

2.2.1.4   Explore with plots

Just for fun, let's plot the data and reproduce Fig. 2A and B. We are using the `qplot` function, which is from the ggplot2 package. Two plots are made and only a subset of the rows are plotted in each (in A, the subset in which playback_treatment=="treat" and, in B, the subset in which playback_treatment=="cont"). This book uses the ggplot2 package extensively.

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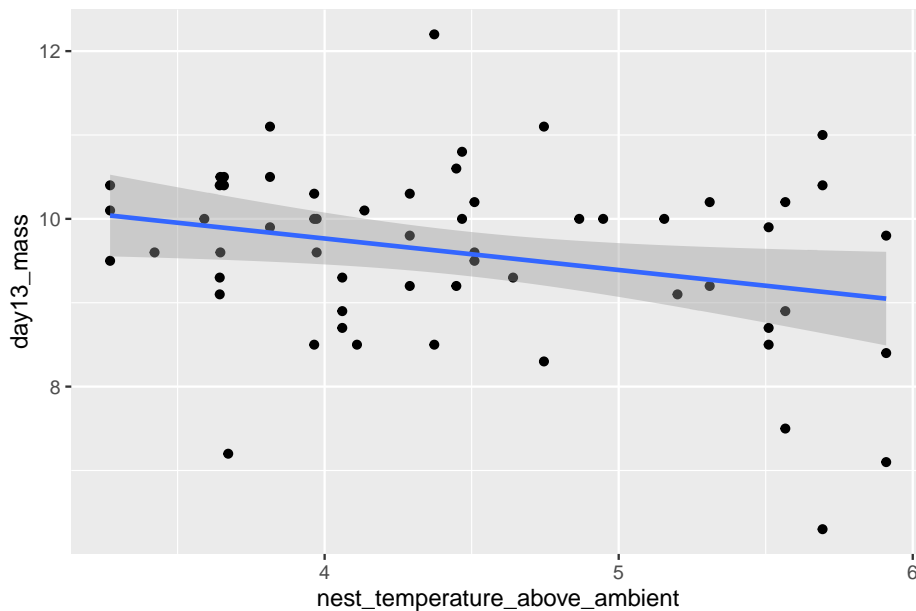


```
qplot(x=nest_temperature_above_ambient, y=day13_mass, data=chick[playback_treatment=="c
  geom_smooth(method="lm")
```
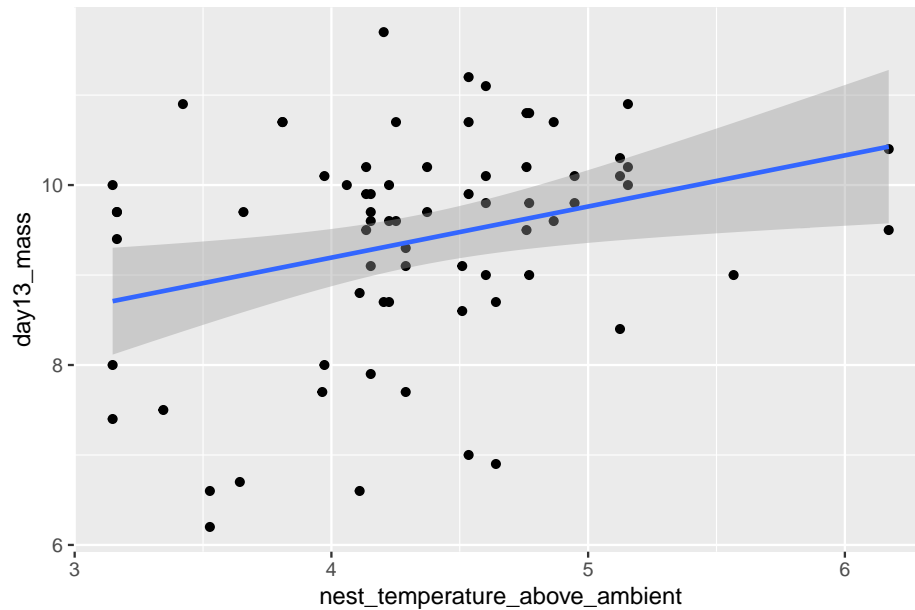
2.2.2 Text file

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"

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

Steps

1. Copy the title of the Dryad page, which is "Data from: Neonicotinoid pesticide reduces bumblebee colony growth and queen production"
2. Create a new folder within "data" and paste in the copied title as the folder name
3. Remove the colon from the name, so the folder name is "Data from Neonicotinoid pesticide reduces bumblebee colony growth and queen production"
4. Download the .csv file into this folder

A .csv file is a text file that is comma-delimted, 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 delimter 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)

```
data_folder <- "Data from Neonicotinoid pesticide reduces bumblebee colony growth and c
filename <- "Whitehorn, O'Connor, Wackers, Goulson (2012) Data from 'Neonicotinoid pest
file_path <- here(data_path, data_folder, filename)
bee <- fread(file_path, header=TRUE)
```

Here, as with the import of the Excel file, the first three lines create the directory path to the file. Peek at the file in the console. Again, there are spaces in the column names. **Here I'll leave it to you to change this**

Here is a reproduction of Fig 2 from the journal article.

```
bee[, treatment:=factor(treatment, c("Control", "Low", "High"))] # reorder factor leve
ggbarplot(data=bee, x="treatment", y="new_queens", add = "mean_se")
```
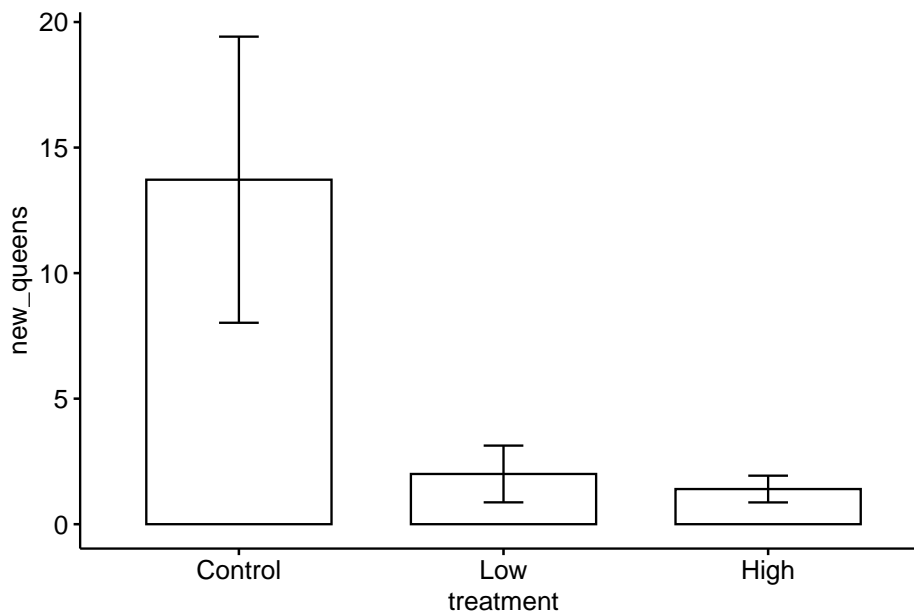

The plot suggests immediately some problems with the plot itself and the associated analysis. First, the y-axis is counts, which means that negative values are impossible. But the standard error bars look like they use standard errors computed from a model that allows infinetly large negative values, and the illustrated standard error bars imply that negative values exist. So these error bars are misleading. Second, it is good practice, especially if sample sizes are

modest or small, to "show the data", which means, show the individual data points and not just a summary of the distribution.

Here are three alternative plots for exploratory purposes. The first simply "shows the data" but still uses the misleading standard error bars. The second uses a box plot. The last plots the means and 95% confidence intervals modeled with a GLM (generalized linear model) to account for the count data (the model used could be improved). Notice that the bar length above the mean is longer than the bar length below the mean (that is the interval is asymmetric about the mean). In order to stay focussed on importing data, I leave explanation of these plots and analysis to later chapters.

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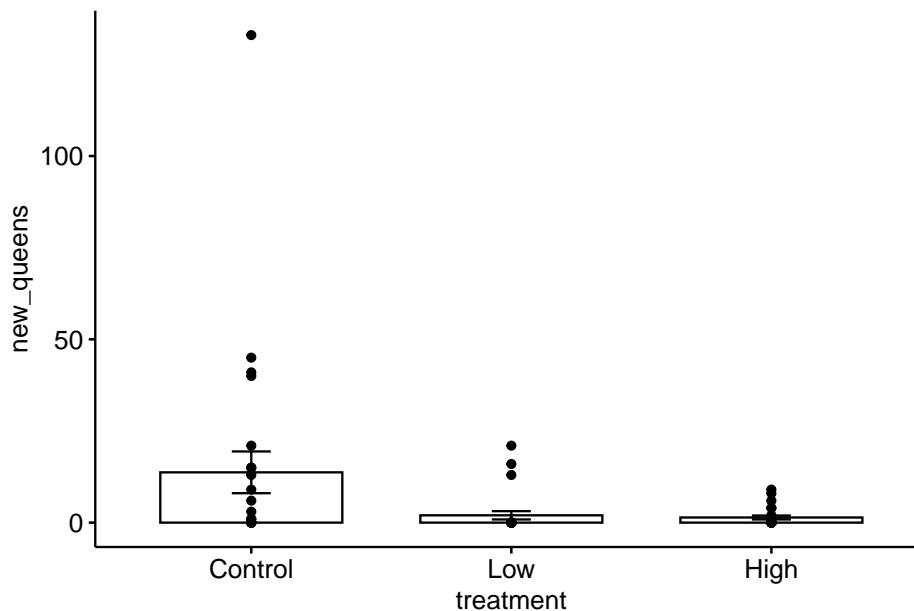


```
ggboxplot(data=bee, x="treatment", y="new_queens")
```
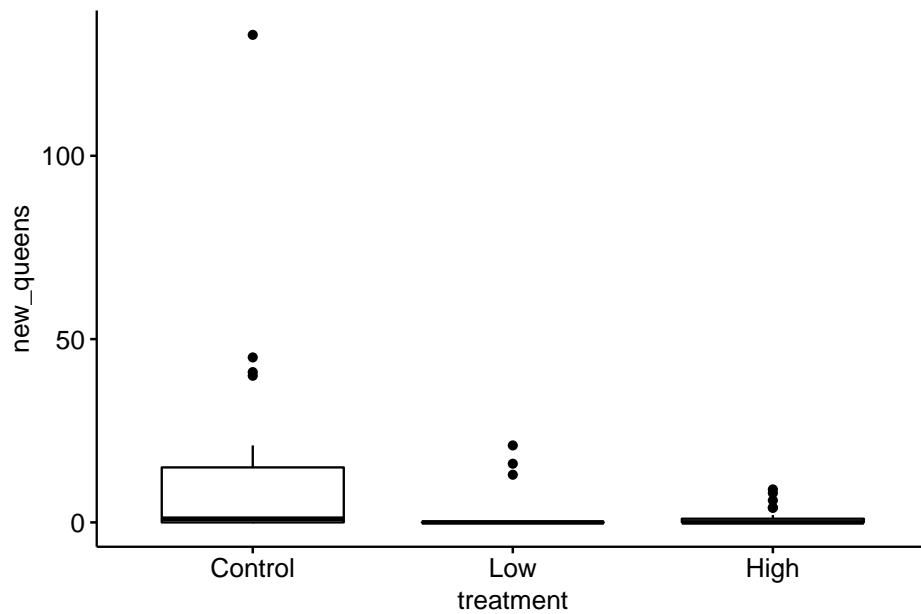
```
fit.glm <- glm(new_queens ~ treatment, data=bee, family=poisson())
means.glm <- emmeans(fit.glm, specs="treatment", type = "response")
gg <- ggplot(data=data.frame(means.glm), aes(x=treatment, y=rate)) +
  geom_col(fill="gray") +
  geom_errorbar(aes(x=treatment, ymin=asymp.LCL, ymax=asymp.UCL), width=0.3) +
  ylab("New queens") +
  NULL
gg
```
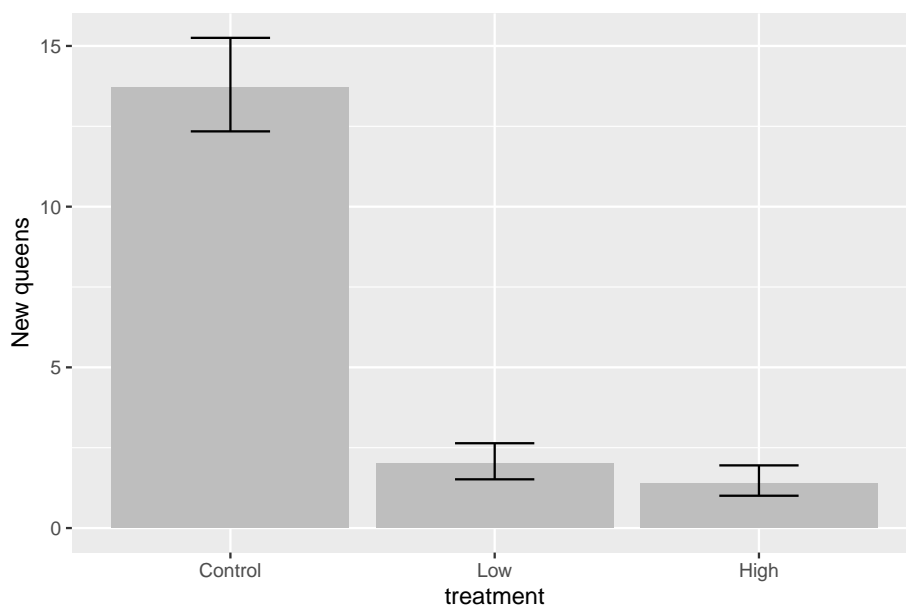
2.3 Data wrangling

2.3.1 Combining data

Source article 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.

Data source https://datadryad.org/stash/dataset/doi:10.5061/dryad.6121hj7

file name: datafiles.xlsx

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.

2.3.2   Subsetting data

It is common to see researchers create multiple subsets of data for further processing. This practice should be 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.

In these examples, dt is the name of a data frame (that has been converted to a data table), not a function. Remember that all functions are followed by parentheses.

1. `dt[sex=="female",]` includes all rows in which the column *sex* contains the value "female"
2. `dt[sex=="female" & age > 18, ]` includes all rows in which the column `sex` contains the value "female" AND the column *age* contains values greater than 18
3. `dt[site=="wells marsh" | site=="scarborough marsh", ]` contains the rows in which the column "site" contains either the value "wells marsh" OR "scarborough marsh"
4. `dt[(site=="wells marsh" | site=="scarborough marsh") & sex == "female", ]` contains the rows in which the column "site" contains either the value "wells marsh" OR "scarborough marsh" AND the column *sex* contains the value "female".

Here I fit a linear model of glucose ~ group to the "12 hour fast" subset of the fasting data that was just imported (*group* is the column containing either "lean" or "obese").

```
m1 <- lm(glucose_t_240_min_m_m ~ group, data=fasting[intervention=="12 h fast", ])
coef(summary(m1))
```

```
##              Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)    4.8375  0.1201747 40.2538890 7.115319e-16
## groupobese     0.0250  0.1699527  0.1470997 8.851507e-01
```

2.3.3   missing data

The function `na.omit` omits all rows with at least one missing value. In general, be vary careful if using `na.omit` because you may delete values that you should be analyzing. For example, here is a fake data set, call it "plant", with two response variables, one of which has a missing value.

treatment

root_mass

shoot_mass

cn

100.2

239.9

cn

205.4

cn

95.6

244.6

cn

149.1

252.1

tr

100.2

239.9

tr

118.0

205.4

tr

95.6

244.6

tr

149.1

252.1

This code "cleans" the data.table *plant* by omitting rows with any missing value

```
plant <- na.omit(plant)
```

The problem is, any analysis of *shoot_mass* will not contain the value 244.6 because it was deleted from the data frame along with the rest of the row containing the missing value.

Instead of deleting the entire row, retain all data but handle missing values at the level of the analysis. This can be done using the subset mechanism. The function `is.na` returns a boolean (a variable with values TRUE or FALSE) for each element in a list. Consequently, the function `!is.na` can be used to obtain the rows in which the value of $root_mass$ is "not na".

```
summary_table <- plant[!is.na(root_mass),
                       .(mean=mean(root_mass),
                         sd=sd(root_mass),
                         se=sd(root_mass)/sqrt(.N),
                         N=.N),
                       by=.(treatment)]
knitr::kable(summary_table, digits=1)
```

treatment

mean

sd

se

N

cn

115.0

29.6

17.1

3

tr

115.7

24.2

12.1

4

Many R functions automatically handle missing values. For example, the `mean` and `sd` functions have NA handling methods. These are used in the script below and generate the correct mean and sd. But the computation of the standard

error fails to compute the correct answer because it uses the wrong N. Compare the results to the script using `!is.na`. Without using `!is.na`, the sample size for the control, *for the variable root_mass*, is incorrect–it is simply the number of rows where *Treatment* is "cn".

```
summary_table <- plant[,
                       .(mean=mean(root_mass, na.rm=TRUE),
                         sd=sd(root_mass, na.rm=TRUE),
                         se=sd(root_mass, na.rm=TRUE)/sqrt(.N),
                         N=.N),
                       by=.(treatment)]
knitr::kable(summary_table, digits=1)
```

treatment

mean

sd

se

N

cn

115.0

29.6

17.1

3

tr

115.7

24.2

12.1

4

Familiarize yourself with these and with the NA handling methods of other functions, especially `lm` and `predict`.

2.3.4 Reshaping data

In general, your data should be in long format. What is long format? Consider data from a simple experiment with a response variable and a treatment variable with three levels. A common way to organize these data in Microsoft Excel is in **wide format**.

Root mass for three treatment levels in wide format

control

extract

charcoal

10.18

4.34

9.57

10.40

3.74

8.59

8.61

4.58

9.69

11.04

4.12

8.07

9.24

2.44

7.63

Wide format is efficient for computations in a spreadsheet, such as computing means and standard deviations of columns, and for plotting. For most statistical analyses in R (and most statistics software), all measures of a variable (such as root biomass in the example above) should be in a single column with a second column identifying the treatment level associated with each measure. This is called **long format**.

Root mass for three treatment levels in long format

treatment

root_mass

control

10.18

control

10.40

control

8.61

control

11.04

control

9.24

extract

4.34

extract

3.74

extract

4.58

extract

4.12

extract

2.44

charcoal

9.57

charcoal

8.59

charcoal

9.69

charcoal

8.07

charcoal

7.63

2.3.4.1 Wide to long

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.

Article: Rolig, A. S., Mittge, E. K., Ganz, J., Troll, J. V., Melancon, E., Wiles, T. J., … Guillemin, K. (2017). The enteric nervous system promotes intestinal health by constraining microbiota composition. PLOS Biology, 15(2), e2000689. https://doi.org/10.1371/journal.pbio.2000689

Data source https://doi.org/10.1371/journal.pbio.2000689.s008

file name: "journal.pbio.2000689.s008.xlsx"

Let's import and tidy the data for figure 2d. Look at the excel file and the data in Fig. 2d. There is a single treament 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.

```
folder <- "Data from The enteric nervous system promotes intestinal health by constrai
filename <- "journal.pbio.2000689.s008.xlsx"

# figure 2D data
sheet_i <- "Figure 2"
range_i <- "F2:I24"
file_path <- here(data_path, folder, filename)
#file_path <- paste(data_path, folder, fn, sep="/")
dt_wide <- data.table(read_excel(file_path, sheet=sheet_i, range=range_i))
# clean names
dt_wide <- clean_names(dt_wide)
# get rid of "_donor"
new_colnames <- c("gf", "wt", "sox10", "iap_mo")
setnames(dt_wide, old=colnames(dt_wide), new=new_colnames)

# wide to long
exp2d <- melt(dt_wide,
              measure.vars=colnames(dt_wide),
              variable.name="donor",
              value.name="count")
```

Look at the wide and long data frames using View. 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. To delete the rows with the missing values, use

```r
# delete rows with no data
exp2d <- na.omit(exp2d)
```

For both analysis and plots, we want to compare values to the control level, which is named "wt" for the exp2d data. That is, we want "wt" to be the *reference* level. To achieve this, the levels of the factor *donor* need to be reordered.

```r
exp2d[, donor:=factor(donor, c("wt", "gf", "sox10", "iap_mo"))]
```

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

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.

data source*: https://doi.org/10.1371/journal.pbio.2003467.s001

```r
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_path, folder, filename)
#file_path <- paste(data_path, folder, fn, sep="/")
dt_wide <- data.table(read_excel(file_path, sheet=sheet_i, range=range_i))
# clean names
dt_wide <- clean_names(dt_wide)

# wide to long
stack_cols <- paste0("replicate", 1:5)
exp_s8a <- melt(dt_wide,
                id.vars=c("media", "time_h"), # strain col not adding any information
                measure.vars=stack_cols,
                variable.name="Replicate",
                value.name="OD600") # measure of absorbance at 600nm
```

2.3.4.2 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

```r
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 `data.table::melt`.

```
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

2.3.5   Miscellaneous data wrangling

2.3.5.1   Vole data

This books uses data on the lifespans of voles given three different treatments
(control, vitamin E supplementation, vitamin C supplmentation), from

Selman, C., McLaren, J.S., Collins, A.R., Duthie, G.G. and Speakman, J.R.,
2013. Deleterious consequences of antioxidant supplementation on lifespan in a
wild-derived mammal. Biology letters, 9(4), p.20130432.

1. Source: Dryad Digital Repository. https://doi.org/10.5061/dryad.31cc4/
   1
2. File: "RSBL-2013-0432 vole data.xlsx"
3. Sheet: "COLD VOLES LIFESPAN"

The vole data were archived in a format that requires some wrangling before
it can be analyzed. The "lifespan (days)" column contains the lifespan for all
three treatment levels, that is, the lifespan values are "stacked" or in "long"
format. What is unusual is that there is no "treatment" column specifying the
three treatment levels (control, vitamin E, vitamin C). Instead there are control,
vitamin E, and vitamin C columns that contain a 1, if the lifespan value belongs
to that treatment, and a NULL value if not. On import, the NULL value is
replaced with NA, to indicate "missing". In order to analyze the data, the three
treatment assignment columns need to be combined into a single "treatment"
column.

```r
folder <- "Data from Deleterious consequences of antioxidant supplementation on lifespan in a wil
filename <- "RSBL-2013-0432 vole data.xlsx"
file_path <- here(data_path, folder, filename)
vole <- data.table(read_excel(file_path, sheet="COLD VOLES LIFESPAN", range="a2:d98"))
# clean column names
vole <- clean_names(vole)

# create treatment column
# in rows where control=1, set treatment value to "control"
vole[control==1, treatment:="control"]
# in rows where vitamin_e=1, set treatment value to "vitamin_e"
vole[vitamin_e==1, treatment:="vitamin_E"]
# in rows where vitamin_c=1, set treatment value to "vitamin_c"
vole[vitamin_c==1, treatment:="vitamin_C"]

# change column "lifespan_days" to "lifespan"
setnames(vole, old="lifespan_days", "lifespan")
```

Plot the data with a stripchart (often called a dot plot) to view the lifespan as
a function of treatment level

```r
ggstripchart(x="treatment", y="lifespan", data=vole,
             add = c("mean_ci"))
```
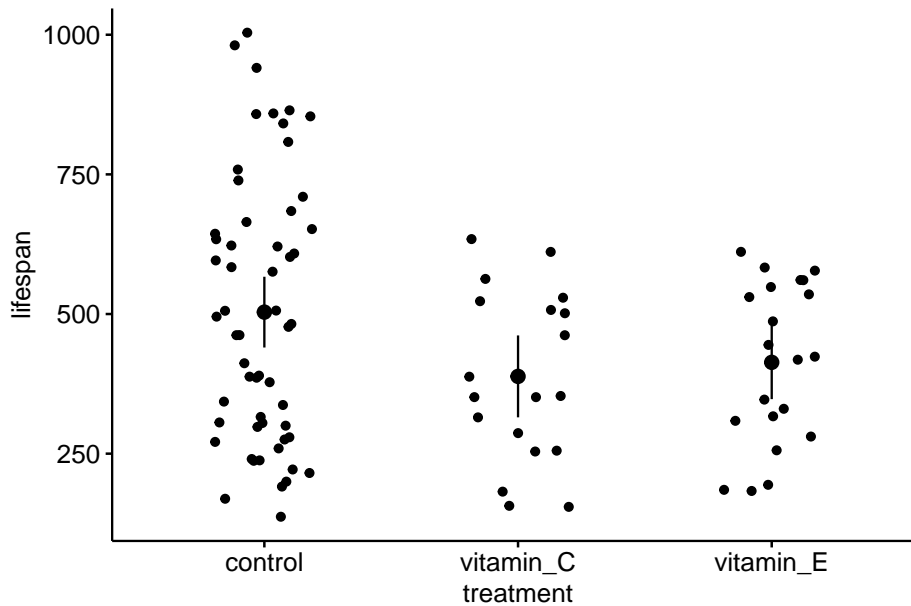
2.4   Saving data

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 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()`

```r
save_file_path <- here(output_path, "vole.Rds")
saveRDS(object = vole, file = save_file_path)

# to read this use
vole <- 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.

```r
# save the data to output folder

# tab delimited
save_file_path <- here(output_path, "vole.txt")
```

```r
write.table(vole, save_file_path, sep="\t", quote=FALSE)

# comma delimited
save_file_path <- here(output_path, "vole.csv")
write.table(vole, 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. "$\widehat{}$" is the standard character string for a tab. Check in your output folder and open the file in a text editor.

2.5 Problems

1. Download the dataset "data-Lodjak.et.al-2016-FuncEcol.xlsx" from the Dryad repository at https://datadryad.org/resource/doi:10.5061/dryad.rd01s. The .xlsx file presents the data cleanly but the trade-off is that the 1) multiple header rows, and 2) spaces in the header labels, 3) parentheses in the header labels make it more complex to import in a usable way. Import the data and plot Body Mass against Age (that is make Body Mass the "Y" variable and Age the "X" variable) using the qplot function. You should recode the column labels to remove spaces and parentheses using the setnames function.

2. Download the dataset "Results2015.txt" from the Dryad repository at https://datadryad.org//resource/doi:10.5061/dryad.65vk4. Try to reproduce Fig. 1. It's not easy. I've inserted the figure below but also download the paper and look at Fig. 1.
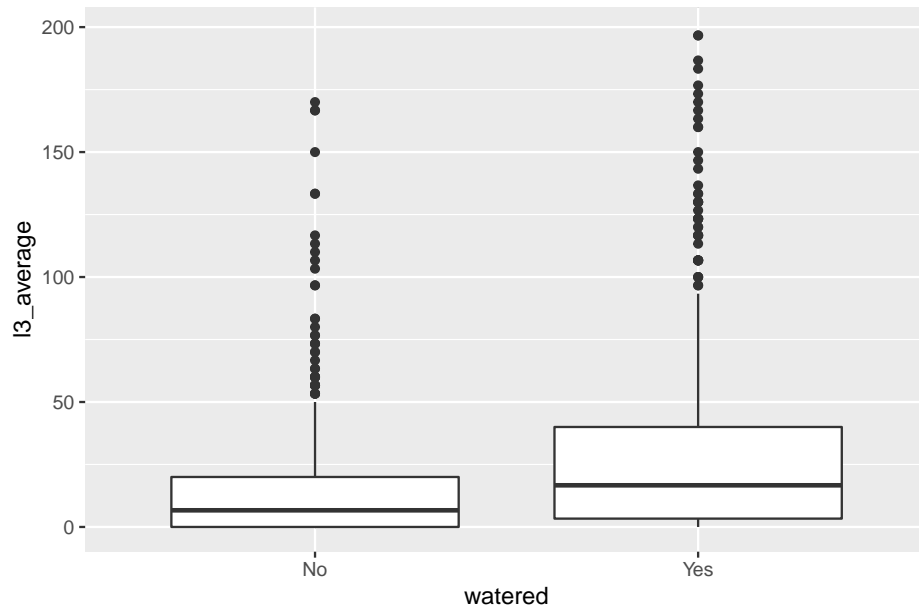
3. (grad students only) Download and plot data from a Dryad Repository dataset of your choice.

Part II: Some Fundamentals of Statistical Modeling

Chapter 3

An Introduction to Statistical Modeling

This chapter introduces statistical modeling using the **linear model**. All students are familiar with the idea of a linear model from learning the equation of a line, which is

$$Y = mX + b \tag{3.1}$$

where m is the slope of the line and b is the Y-intercept. It is useful to think of equation (3.1) 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 (3.1), 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 3.1A is just that, a line, but the line in Figure 3.1B is a linear model fit to the data in Figure 3.1B.

3.1 Two specifications of a linear model

3.1.1 The "error draw" specification

A linear model is commonly specified using

$$Y = \beta_0 + \beta_1 X + \varepsilon \tag{3.2}$$

$$\tag{3.3}$$

Figure 3.1: 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.

This specification of a linear model has two parts: the **linear predictor** $Y = \beta_0 + \beta_1 X$ and the **error** ε. The linear predictor part 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.

As with the equation for a line, the linear predictor part 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** given a specific input value of X. This is often written as $\mathrm{E}[Y|X]$, which is read as "the expected value of Y given X", where "given X" means a specific value of X. Importantly, $\mathrm{E}[Y|X]$ is the **conditional mean**, which is the *modeled* value of Y for all observations in which X takes some specific value x.

Introductory textbooks almost always introduce linear models using equation (3.3) above. The key part of the model that is missing from the specification above is a second line

$$\varepsilon \sim N(0, \sigma^2) \tag{3.4}$$

which is read as "epsilon is distributed as Normal with mean zero and variance sigma squared". This line explicitly specifies the distribution of the error part. The error part of a linear model is a random "draw" from a normal distribution with mean zero and variance σ^2. Think of this as adding some random value to the expected value.

3.1.2 The "conditional draw" specification

A second way of specifying a linear model is

$$y_i \sim N(\mu_i, \sigma^2) \tag{3.5}$$
$$\mathrm{E}(Y|X) = \mu \tag{3.6}$$
$$\mu_i = \beta_0 + \beta_1 x_i \tag{3.7}$$

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 conditional 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.

3.1.3 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 varible Y) were generated. The error draw specification "generates" data by randomly drawing some error ε_i from the specified distribution and adding this to x_i. The conditional draw specification "generates" data by constructing what y_i "should be" given x_i (the conditional expection), and then drawing a random variable from a distribution with this expectation. This random draw is y_i and not the "error". 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.

The conditional draw specification explicitly defines all parameters, including the parameters of the linear predictor (β_0 and β_1), the conditional mean μ and the variance σ^2. The error draw specification only defines the parameters of the linear predictor, and often these are referrred to as "the parameters" in the sense that there are not other parameters. The error draw specification is most useful for thinking about model checking a fit linear model. The random draw

specification is more generally useful in that it is easily generalized to more complex models, including hierarchical models, generalized linear models, and Bayesian models. In fact, *thinking about a model as a predictor plus error can lead to the misconception that in a generalized linear models, the error has the distribution (binomial, poisson, etc.) modeled.*

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.

3.2   What do we call the X and Y variables?

The inputs to a linear model (the X variables) have many names including "independent variables," "predictor variables,", "explanatory variables," "treatment variables," and "covariates". The output of a linear model (the Y variable or variables if the model is multivariate) is the "dependent variable," "response," or "outcome." The β in the linear model are model **parameters** and if a parameter is multiplied by an X variable then it is also a **coefficient** (for example, β_1 in model (3.3) is a coefficient). The coefficients of the X in a linear model (β_1 in model (3.3)) are often called "the effects" (so β_1 is the effect of X_1).

3.3   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 enviromentally 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.

3.4 Modeling strategy

1. **choose a model**. 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 (18.7).

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 data reasonably approximate the chosen model.

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 estimated parameters (or other results dervived from the estimates) with their uncertainty.

In order to use a statistical model to describe, predict, or explain, we need to fit a model to data in order to estimate the parameters. A linear model fit to some data is

$$\hat{y}_i = b_0 + b_1 x_i + e_i \tag{3.8}$$

$$\tag{3.9}$$

$\hat{y}_i$ ("y hat") is the **predicted value** of individual i, b_0 and b_1 are the coefficients of the model fit (though technically b_0 is not a coefficient), and e_i is the residual. Sometimes $\hat{y}_i$ is simply called "the prediction".

If our goal is inference – to infer something about the "population" from the sample using the fit model, then $\hat{y}_i$ is the **point estimate** of the parameter μ_i, the coefficients b_0 and b_1 are point estimates of the parameters β_0 and β_1, and the standard deviation of the e_i is an estimate of σ. "Population" is in quotes because it is a very abstract concept. Throughout this book, Greek letters refer to a theoretical parameter and Roman letters refer to point estimates.

Throughout this text, I recommend reporting and interpreting **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).
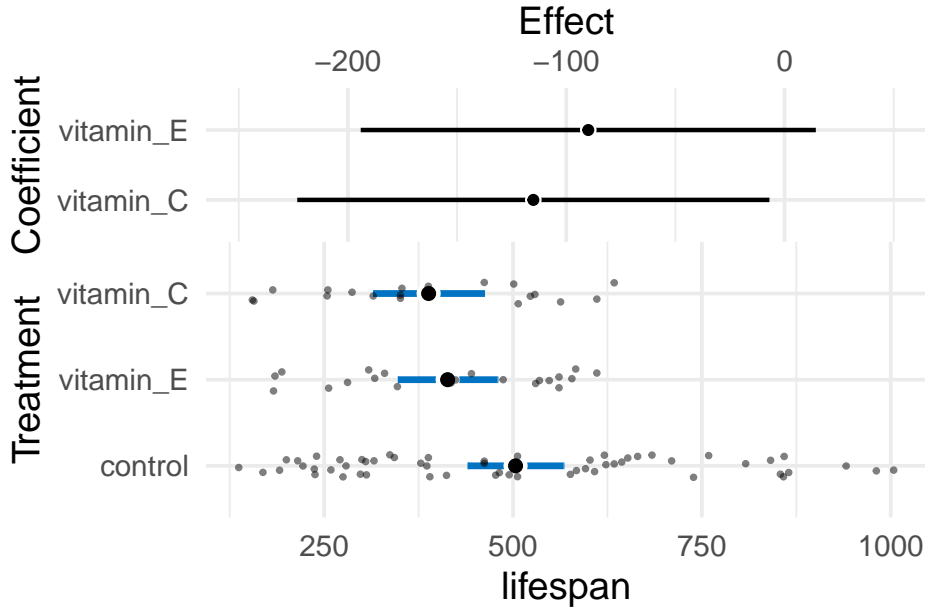
Figure 3.2: HarrellPlot of vole data.

For the model fit to the data in Figure 3.1B, the coefficient of X is the slope of the line. Perhaps surprisingly, we can fit a model like equation (3.3) to data in which the X variable is categorical. A simple example is the experiment of antioxidants (vitamins C and E) on lifespan in Voles (Fig. 3.2). In this experiment, the X variable is categorical, with three **levels**: "Control", "Vitamin_E" and "Vitamin_C". Categorical X variables are often called **factors**. The trick to using a statistical model with categorical X is to recode the factor levels into numbers – how this is done is explained in Chapter xxx. When the X variable is categorical, the coefficients of the X are *differences in group means*. The linear model fit to the vole data has two coefficients, one for Vitamin E and one for vitamin C. The estimate and uncertainty of the these two coefficients are shown in the top part of Figure 3.2. The bottom part shows the raw data, as well as the group (factor level) means and the uncertainty in the estimate of these means.

3.5 A mean is the simplest model

The simplest possible model that can be fit to the data is

$$\mathrm{E}[Y] = b_0 \tag{3.10}$$

which is simply the mean of Y, or, more specifically, the **unconditional mean** of Y, since its value is not conditional on any value of X.

3.6   Assumptions for inference with a statistical model

Inference refers to using the fit model to generalize from the sample to the population, which assumes that the response is drawn from some specified probability distribution (Normal, or Poisson, or Bernouli, etc.). Throughout this text, I emphasize reporting and interpreting point estimates and confidence intervals. 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. A significance test results in a **p-value**, which can be reported with the point estimate and confidence interval. Somewhat related to a significance test is a hypothesis test, or what is now often perjoratively called 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. NHST may be useful for some very limited kinds of science but, in general, is not useful for most biological research and, instead, leads to large misconceptions. A general rule of thumb is, do not compare a reported p-value to α.

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 (3.3) and (18.7) above. For example, a cubic polynomial model

$$\mathrm{E}(Y|X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \tag{3.11}$$

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

$$\mathrm{E}(Y|X) = \mathbf{X}\boldsymbol{\beta} \tag{3.12}$$

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

$$\eta = \mathbf{X}\boldsymbol{\beta} \tag{3.13}$$

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

$$E(Y|X) = \frac{\beta_1 X}{\beta_2 + X} \tag{3.14}$$

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** observations (the Y), that is, for any two observations with the same value of X, we cannot predict the value of one given the value of the other. For example, in the vole data above, uncorrelated implies that we cannot predict the lifespan of one vole within the Vitamin E treatment given the lifespan of another vole in the Vitamin E treatment. For linear models, this assumption is often stated as "independent errors" (the ε in model (3.3)) instead of independent observations.

There are lots of reasons that observations might be correlated. In the vole example, perhaps the voles were housed in batches of 5 individuals, and slight differences in the environment among the housing containers, caused all the voles in some containers to have shorter lifespans than expected given their treatment assigment and all voles in other containers to have longer lifespans than expected given their treatment assigment. 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 (3.3). 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 (3.3).

3.7   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 (3.3) are all from the same (or **identical**) distribution. This is explicitly stated in the second line of model specification (3.3). If we were to think about this using model specification (18.7), 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 (18.7), we'd say the conditional probablity 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 mu_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 (sucessful escape or sucessful infestation of host) are not continuous and often often have asymmetric probablity 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 (18.7).

A common misconception is that inference from a linear model assumes that the *response* (Y) is normally distributed. Both the "linear model" and "statistical model" ways of specifying the model show precisely why this conception is wrong. Model (3.3) 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 (18.7) states that the conditional outcome has a normal distribution, that is, the distribution after adjusting for variation in X.

3.8 "Statistical model" or "regression 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. Other patterns occur. For example "ANCOVA models" often include interaction effects but "regression models" rarely do. To avoid thinking of statistical analysis as "regression vs. ANOVA" (the type of thinking encouraged by many textbooks in biostatistics), I will most often use the term "statistical model" for general usage, and use a more specific term only to emphasize something about the model in that particluar context.

3.9 GLM vs. GLM vs. GLS

Linear models are sometimes called "general linear models" with the abbreviation GLM. This is unfortunate because the abbreviation GLM usually refers to **generalized linear models**. Regardless, don't confuse either version of GLM with GLS, which is the abbreviation of **generalized least squares**. GLS generalizes the linear model to allow for heteroskedastic and/or correlated error (using the "linear model" way of thinking about model specification)

Chapter 4

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.

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

4.1.1 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
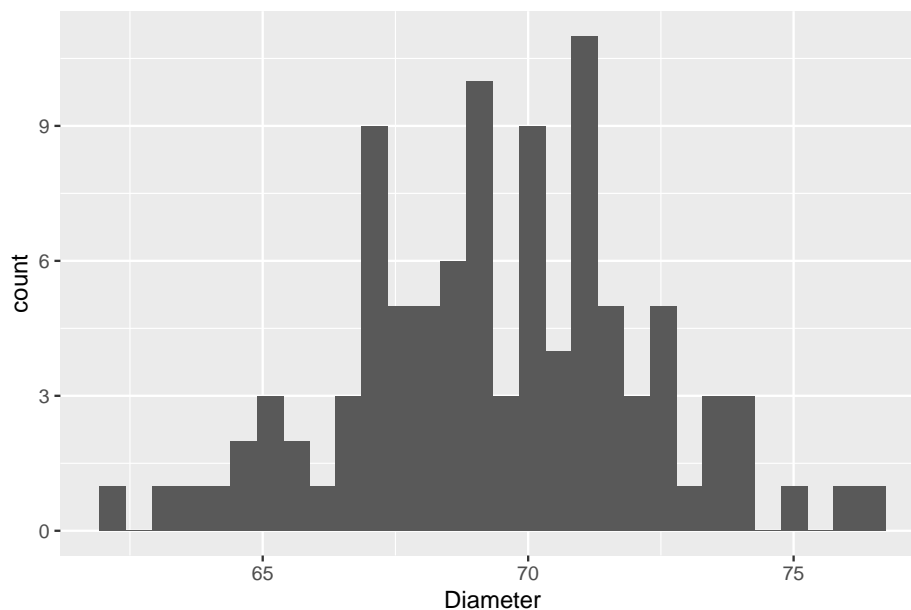
The mean of this sample is 69.4 and the standard deviation is 2.8. The standard deviation is the square root of the variance, and so computed by

$$s_y = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \overline{y})^2}{n - 1}} \tag{4.1}$$

Memorize this equation. To understand the logic of this measure of variability, note that $y_i - \overline{y}$ is the **deviation** of the ith 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).

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 interpet.
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.

4.1.2 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 smaller the standard error, the more precise the estimate. The standard error of the mean (SEM) is computed as

$$SEM = \frac{s_y}{\sqrt{n}} \tag{4.2}$$

The SEM is often denoted $s_{\bar{y}}$ to indicate that it is a standard deviation of the mean ($\bar{y}$). In what sense is a standard error a measure of variability? 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 1) resampling 100 cells and 2) recomputing a mean from the re-sampled data, then repeating this resampling and recomputation an infinite number of times and each time, you write down the newly computed mean. The true standard error of the mean is the standard deviation of this infinitely long column of means. This means that a standard error of the mean, computed from a single sample using equation (4.2) is itself a sample statistic.

Notes on standard errors

1. 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 (4.2). For other statistics, a computer intensive method such as the **bootstrap** is necessary

to compute a standard error. We will return to the bootstrap at the end of this chapter.

2. The units of a standard error are the units of the measured variable.

3. 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.

4. This last point (the SEM decreases with sample size) seems obvious when looking at equation (4.2), since n is in the denominator. Of course n is also in the denominator of equation (4.1) 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 (4.1). The standard deviation is the square root of an average and averages don't increase with the number of things summed since both the the numerator (a sum) and denominator increase with n.

4.2   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.

4.2.1   Steps

1. Open Google Sheets
2. In cell A1 type "mu". mu is the greek letter μ and is very common notation for the poplation 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 to the value in the cell above, so the resulting value should be 2.

6. In Cell B8, insert the equation "=normsinv(rand())*B2 + B1". 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 to "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.

4.3 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)

4.3.1 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 errror 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*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?

4.3.2   part II - means

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

```
qplot(means)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```
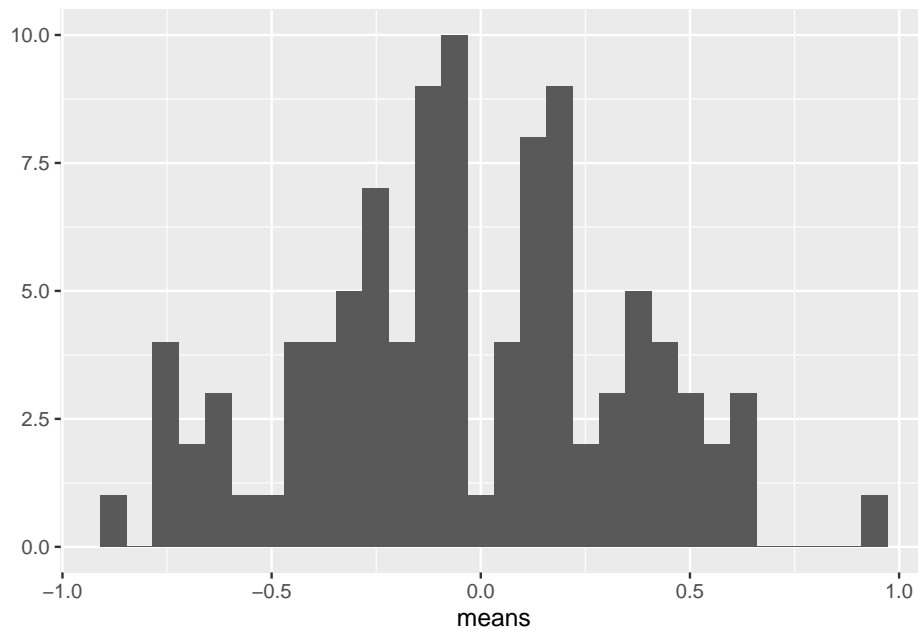
Compute the mean of the means

```
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, then the mean estimated with $n = 6$?
4. Redo with $n = 10000$

4.3.3   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

```r
mean(sems)
```

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

Congratulations, you have just done a Monte Carlo simulation!

4.3.4 Part IV – Generating fake data with for-loops

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

```r
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
```

```r
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?

4.4 Bootstrapped standard errors

A standard error of the mean is the expected standard deviation of an infinite number of hypothetically re-sampled means. A bootstrap standard error of a statistic is the standard deviation of the statistic from a finite number of *resamples* of the data.

Let's download some data to explore this concept. The data are archived at Dryad Repository.

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.

I've read in the file using read_excel and converted to a data.table named `vole`. I used `setnames` to rename the columns to lifespan, control, vitamin_E, and vitamin_C. The data are in a **wide format** – that is instead of a single "treatment" column, there are three columns ("control", "vitamin C", "vitamin E") with value = 1, if that row (or lifespan) was assigned the treatment of the column label and zero otherwise. In general, we want data.tables to be in long format.

Compute the standard error of the mean of the lifespan for the control group using equation (4.2). One simple way to do this for the control group is to extract the subset of the data satisfying the condition control = 1 (the value in the column "control" equals 1). In R, these conditional querries use `==`.

```
control_voles <- na.omit(vole[control==1, lifespan]) # subset of data satisfying condi
# omitting missing data, if these exist
n <- length(control_voles) # the sample size
se <- sd(control_voles)/sqrt(n-1) # standard error of the mean of the control voles
```

Okay, the SEM using equation (4.2) is 31.9. Let's compare this with a bootstrap estimate of the SEM. There are several R packages that do a bootstrap but here I've coded one using a for-loop both to practice with for-loops and to help communicate what a bootstrap *is*.

The basic algorithm for a bootstrap is

1. **re-sample the data with replacement**. Here "the data" is the set of lifespans for the Control voles. "Resample with replacement" means to sample n times from the full set of values. If we were to do this manually,

we would i) write down each value of Control lifespan 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).

2. Compute the statistic with the new sample.
3. Repeat steps 1 and 2 p times, which will result in a p-length vector of the statistic. Typically p is somewhere between 2000 and 10,000.
4. For the standard error of the statistic, compute the standard deviation of the re-sampled statistic.

A for-loop automates this algorithm.

```r
set.seed(1)
n_iter <- 2000 # number of bootstrap iterations, or p
means <- numeric(n_iter) # we will save the means each iteration to this
inc <- 1:n # the first sample is the actual sample
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_boot <- sd(means)
getwd()
```

```
## [1] "/Users/jwalker/Documents/Github projects/Bookdown projects/applied-biostats"
```

The SEM using the bootstrap is 31.43. Again, the SEM using equation (4.2) is 31.9.

4.5 Confidence Interval

Here I introduce a **confidence interval** (abbreviated to CI) using the mean of a sample but the concept is easily generalized to any parameter of a statistical model. The mean of the Control voles is 503.4 and the SE of the mean is 31.9. The SE is used to construct the lower and upper boundary of a "1 - α" confidence interval using `lower <- mean(x) - qt(alpha/2, df = n)*se(x)` and `upper <- mean(x) + qt(alpha/2, df = n)*se(x)`.

$$upper = mean+ \tag{4.3}$$

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. 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 true mean lies within the interval". Perhaps a more useful interpretation is that the interval contains the range of means that are consistent 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).

Chapter 5

Covariance and Correlation

Variance is one of two major concepts in statistical modeling. The second is **covariance**, which arises when two variables measured on the same unit vary together. "Vary together" means that if we measured leg length and arm length on each individual in a sample of humans, we'd expect individuals with long arms to also have long legs while those with short arms to have short legs. "Measured on the same unit" means that we measure both leg length and arm length in each individual of a sample – we cannot compute a covariance if we measure legs in one sample and arms in a second. Covariance can be positive or negative. It is positive when the tendency is for both values to be large or both values to be small. It is negative when the tendency is for one value to be small when the other is large. Positive and negative covariance are easily visualized with a scatterplot 5.1.

1. Covariance

$$\text{COV}[X, Y] = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n - 1} \tag{5.1}$$

Compare this to the equation for the variance. In (5.1), the numerator in the sum is the product of two different deviations (one for each variable) instead of the product of a deviation with itself.

2. Correlation

3. Regression
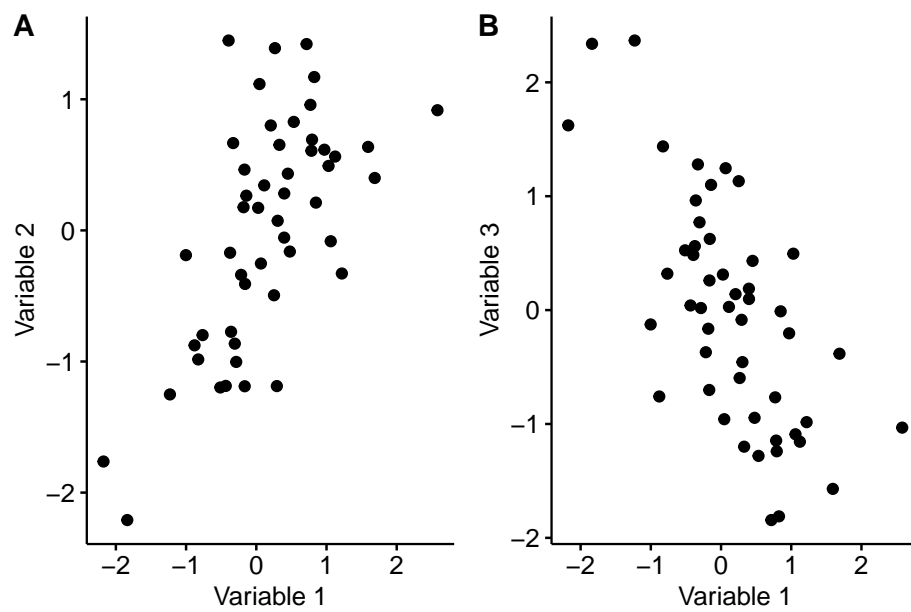
Figure 5.1: Scatterplot illustrating two variables with (A) positive covariance and (B) negative covariance

Chapter 6

P-values

6.1 *p*-values

Let's use the vole data to introduce the *p*-value. The archived format of these data requires some wrangling – a method for this is in Vole data.

A typical analysis would compare the means of the two supplement treatment levels with the control treatment level and use a *p*-value as evidence of "an effect".

contrast

estimate

SE

df

lower.CL

upper.CL

t.ratio

p.value

vitamin_C - control

-115.1

54

93

-223.2

-6.9

-2.1

0.037

vitamin_E - control

-89.9

52

93

-194.1

14.3

-1.7

0.090

vitamin_E - vitamin_C

25.2

65

93

-103.8

154.1

0.4

0.699

The table above gives the SE, t and p-value for each pairwise contrast (difference in means) among the three treatment levels. A typical report (one with several misconceptions) might read

"We found a significant effect of Vitamin C (t = -2.1, p = 0.037) on lifespan, but no effect of vitamin E (t = -1.7, p = 0.09) on lifespan."

A p value *is a continuous measure of evidence against the null.* As long as the data approximate the assumptions of the null hypothesis pretty well, a very small p-value, such as 0.0005, is pretty good evidence against the null hypothesis – but does not mean "an effect exists". To show an effect exists, we should have small p-values in multiple replicates *and* we should rigorously probe the hypothesis with different experiments that challenge the hypothesis in different ways. A small p is evidence for a research program to move forward with replication and probing. A big p-value, say 0.22 or 0.76, is pretty weak evidence against the null, but does not mean "there is no effect." If an experiment is well designed, a big p could suggest abandoning any hypotheses that predict biologically consequential effects. Unfortunately, a big p could also reflect a weak experimental design. Between small and big p values, such as 0.009 or 0.11, problems arise. These intermediate p-values beg for replication. A major

problem of inference using p values is that there is no sharp boundaries in how to interpret p-values. Biologists, encouraged by both the literature and textbooks, typically use $p < 0.05$ as a sharp boundary to declare that an effect exists or not. This "dichotomania" is discussed below.

Okay. so what *is* a p-value? 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 $P(data|null)$, 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 $\text{prob}(t \geq t_{obs}|H_0)$. 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.

6.2   Creating a null distribution.

Let's repeat: 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

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

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 vitamin E effect on lifespan, then the expected difference between the means of the control and vitamin E treatment levels is zero. Or,

$$\mathrm{E}(\bar{vitamin}_E - \bar{control}|H_o) = 0.0 \tag{6.2}$$

let's plot the data and look at the group means. Below is a strip chart of the vole data with superimposed treatment level means, using the function `ggstripchart` from the ggpubr package (can you make this?). I'm going to refer to this kind of chart as a "dot plot", which is what most biology researchers call this type of chart.
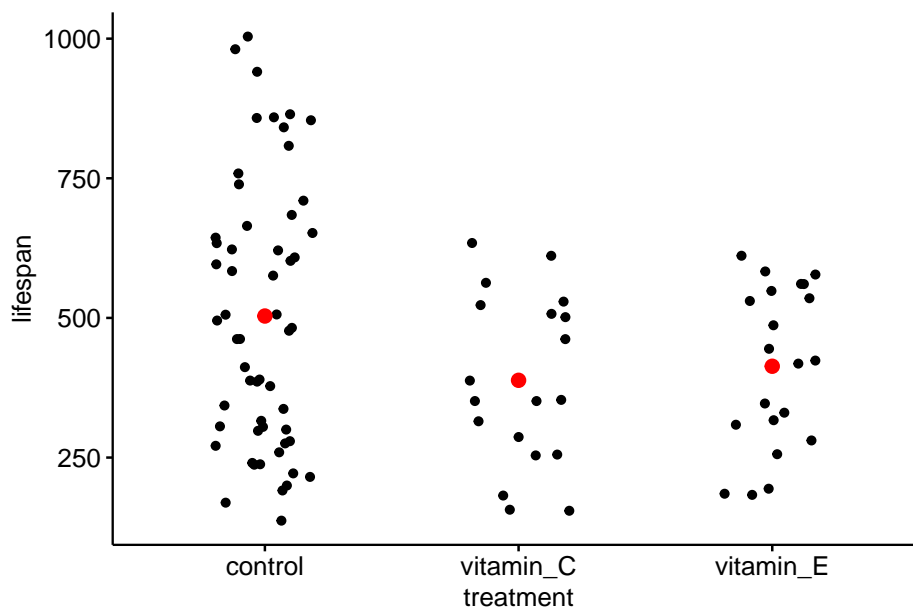
6.2.1 the Null Distribution

The mean lifespan in the vitamin_E treatment is 89.9 days shorter than the mean lifespan in the control treatment. And, the mean lifespan in the vitamin_E treatment is 115.1 days shorter than the mean lifespan in the control treatment. These are the measured effects, or the **observed differences in means**. How confident are we in these effects? Certainly, if the researchers did the experiment with *two* control 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 lifespan 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
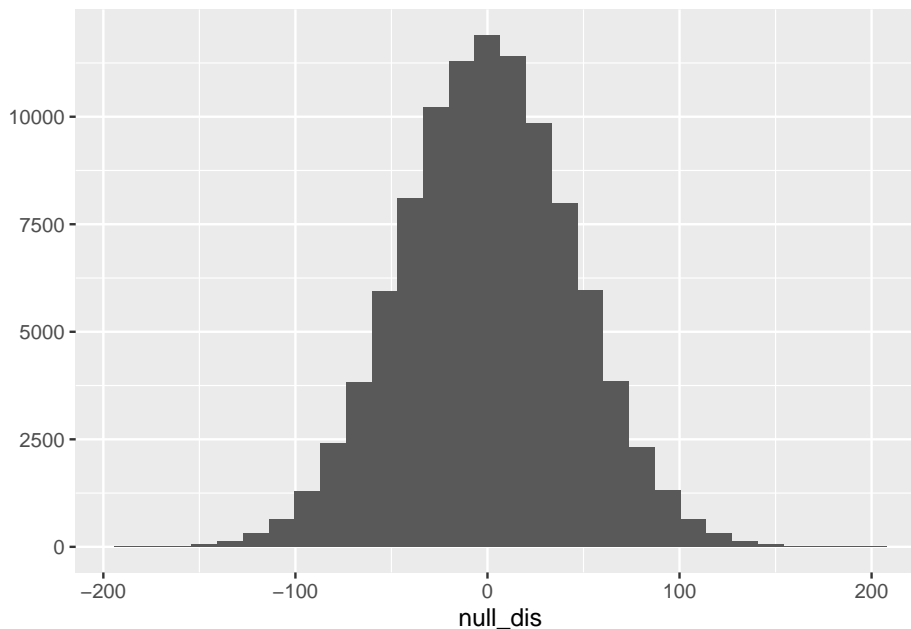
Figure 6.1: Null distribution for an infinitely large data set that looks curiously like the lifespans of the cold-rear voles from the control treatment.

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 am generating a null distribution that is relevant to the cold vole data. See if you can understand the script before reading the explanation below.

```
seed <- 1
n_iter <- 10^5 # number of iterations
mu <- mean(vole[treatment=='control', lifespan])
sigma <- sd(vole[treatment=='control', lifespan])
n <- nrow((vole[treatment=='control',]))
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
null_dis <- apply(sample2, 2, mean) - apply(sample1, 2, mean)
qplot(null_dis)
```

What have we done above? We've simulated an infinitely large population of voles that have a distribution of lifespans similar to that of the cold-reared voles

assigned to the control group. The mean μ and standard deviation σ of the simulated lifespan are equal to the observed mean and standard deviation of the lifespans of the control voles. Then, the script:

1. randomly sample 56 values from this population of simulated lifespans and assign to sample1. We sample 56 values because that is the sample size of our control in the experiment.
2. randomly sample 56 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 (-0.11584) 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 +100 and -100. What this means is that it would be rare to randomly sample two sets of data from the same population with mean μ and standard deviation σ and find a difference of, say, -257. In fact, in the 100,000 runs, there were no difference as large as |-257| (the absolute value of -257). The minimum and maximum differences sampled over the 100,000 iterations was -187 days and 201 days.

How do our observed differences compare? Let's focus on vitamin E. The vitamin_E effect is -89.9 days. There are 2110 sampled differences less than the observed value and 2126 greater than the absolute value of the observed value. Together this is 4236 so the frequency of differences from the simulated null distribution that as larger or larger than the observed difference is 0.042 (this compuation includes the observed value in both the numerator and denominator).

6.2.2   t-tests

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 a value (often zero)

A *t*-test compares an observed *t*-value to a *t*-distribution. The null distribution introduced above was a distribution of mean differences. This isn't generally useful, since the distribution of expected mean differences under the null will be unique to every study. A *t*-distribution is a distribution of *t*-values under the

null (statistical jargon for "given the null is true"), where a t-value is a difference standardized by its standard error. For a two-sample t-test, this is

$$t = \frac{\bar{X}_1 - \bar{X}_2}{SE_{\bar{X}_1 - \bar{X}_2}} \tag{6.3}$$

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 eq xxx), 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 2 are unlikely if the null is true.

The difference between the mean of the vitamin_E treatment and the control treatment is -89.9. A two-sample t-test of this difference is 0.1078146

The p-value comes from comparing the observed t to a null t distribution and "counting" the values that are bigger than the observed t. These are counted in both tails, because p is the probability of a t more extreme than the observed value, and t can be more extreme in the negative direction and in the positive direction. We can simulate this with a finite, instead of infinite, null distribution using the t-distribution instead of the distribution of mean differences, as above. I show the script, but don't just cut and paste the code. Spend time thinking about what the each line does. Explore it by copying parts and pasting into console.

```
set.seed(1)
n_iter <- 10^4 # number of iterations
mu <- mean(vole[treatment=='control', lifespan])
sigma <- sd(vole[treatment=='control', lifespan])
n <- nrow((vole[treatment=='control',]))
sample1 <- matrix(rnorm(n*n_iter, mean=mu, sd=sigma), nrow=n) # 100,000 samples
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 fu
se_mean_diffs <- sqrt(apply(sample2, 2, sd)^2/n + apply(sample1, 2, sd)^2/n)
t_dis <- mean_diffs/se_mean_diffs
```
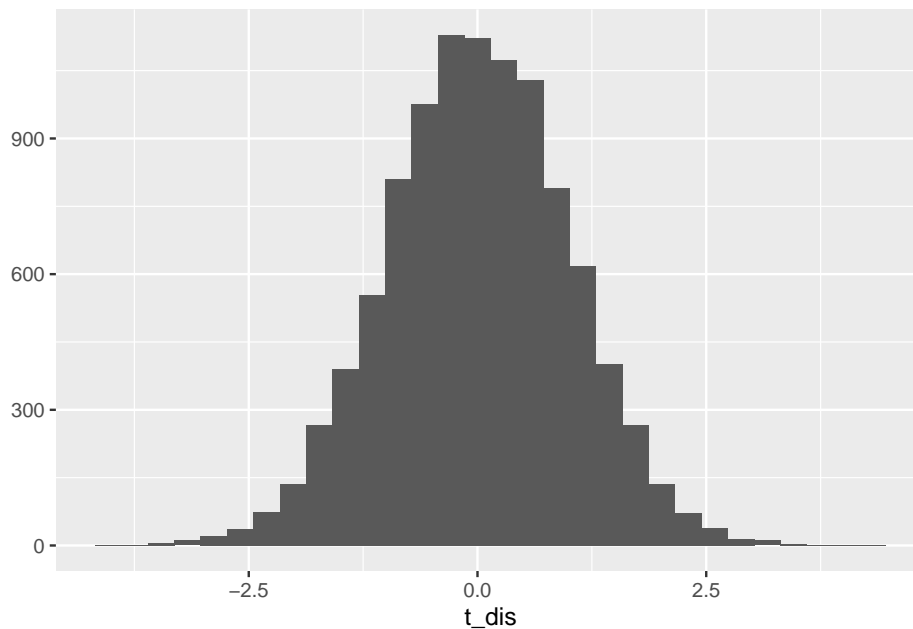
Figure 6.2: Null distribution of t-values. The simulation generated 10,000 t-tests with a true null.

```
#way no.2 - compute the t-tests using the base R function t.test
# rbind stacks sample1 under sample2. I used this order to be consistent with
# t.test function
t_dis2 <- apply(rbind(sample2, sample1), 2, function(x)
  {t.test(x[1:n], x[(n+1):(2*n)], var.equal=TRUE)$statistic})
# confirm that t_dis = t_dis2

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

```
# 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 compute the observed t-value
t_obs <- t.test(x=vole[treatment=='vitamin_E', lifespan],
       y=vole[treatment=='control', lifespan],
       var.equal=TRUE)$statistic
```

```
# 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)) + 1

# 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
(p_vitamin_E <- count/(n_iter + 1))
```

[1] 0.1014899

Hey that looks pretty good! A p value can be computed by counting the number of simulated t-values, *including the observed value*, that are more extreme (in either the positive or negative direction) than the observed t. Including the observed t, there are 1015 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 0.1014899. This simulation-based p-value is very (very!) close to that computed from the observed t-test.

6.2.3   P-values from the perspective of permutation

A very intuitive way to think about p-values is with a permutation test. Consider two of the treatment levels in the vole data, say vitamin E and the vitamin C (I'm bored with the control!). Think about the structure of the dataset: there are two columns, "Treatment", which contains the assigned treatment, and "Lifespan". The values in the Treatment column were randomly assigned prior to the start of the experiment. If there is an effect of treatment on lifespan, then assginment matters – the values in the lifespan column for the vitamin E rows will be more or less, on average, than the values in the lifepan column for the vitamin C rows. Or, the lifespan values are what they are *because* of the values in the treatment column.

Now let's leave the values in the treatment column be, and just randomly re-arrange or permute the lifespan values. What is the new expected diference in lifespan between the two treatments? Zero, of course! That is, because the lifespans 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 lifespan column. A difference in means can be computed from each of these permuations 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 distributin of the statistic computed over many thousands of permutations.

6.3 Statistical modeling instead of hypothesis testing

This chapter is an introduction to a *p*-value by way of *t*-tests. I advocate that you analyze *t*-test like questions using statistical modeling instead of null hypothesis significance testing. The reason is that we learn much more from an estimate of the effect and a CI than from a *t* and *p*-value. But, it is also good to know that a *t* test is a special case of a linear model, and you can get that *t* and *p* using a statistical modeling approach should your boss want them (and you cannot convince them otherwise). Let's explore this.

1. Using the emmeans package, compute the effects (differences in means) of vitamin E and vitamin C on lifespan, relative to the control, with their 95% CI and the *t* and *p* values for the cold-reared vole data.
2. Compute a separate *t*-test of vitamin-E vs. control and vitamin C vs. control.

Are the *t* and *p* values the same? No! The reason is that the statistical model had three groups and the SE of the difference was computed from the sample standard deviation of all three groups. Each t-test computes the SE of the difference from only the two groups being compared. In general, the SE computed from all three groups is better because it uses more information. This is one reason to prefer the linear model instead of the separate t-tests.

3. To convince yourself that a *t*-test is a special case as of a linear model, compute the effects of the vitamin E treatment (relative to control) **but exclude the vitamin C data from the model fit**. Now compare the *t* and *p* values with the *t*-test. These should be the same.

4. Now use the default t.test function by deleting "var.equal=TRUE" from the function. Are *t* and *p* still equal to those from the statistical model? No! the reason is because the default t.test function uses a modification of the t-test called "Welsch's t-test". This test allows for heterogenity of variances. Several sources argue that one should always uses Welsch's test since it simplifies to the classical t-test when the sample variances are equal. This is true, but only relevant if you're into *t*-tests. And, we can model heterogenous variances using a statistical model. We'll do this in a later chapter.

5. Use the function `pairwise.t.test` to compute all pairwise t.tests among the three treatment levels. Is the *p*-value for the vitamin_E - control contrast the same as that if using t.test (with var.equal=TRUE) or the statistical model with vitamin_C data excluded? No! The reason is that pairwise.t.test adjusts the p-values for multiple testing as a default.

Pro tip: Before you use a new R function like t.test or pairwise.t.test, it is really advisable to read the help page and look at the defaults for the parameters!

Researchers publish errors because they failed to look closely at what the R function was doing and they think the function is doing something else. Ooops!

6.4   frequentist probability and the interpretation of p-values

6.4.1   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, but 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 continous outcomes, a frequentist probability is the long run frquency of *observing the outcome or one more extreme.*

6.4.2   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 $P(data|null)$, 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 $\text{prob}(t \geq t_{obs}|H_0)$. 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.

6.4.3   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.

6.4.3.1   Fisher's interpretation

Fisher was working in the context of an agricultural field station, the goal of which is 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 better the evidence that the means 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.

6.4.3.2   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 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. This is the balance. We want to make sure that we test our batch of beer using enough cans to find a bad batch if it exists, but we don't want to test too many cans because this is a waste of money. An experimenter sets α, computes the sample size needed to achieve a certain level of power $(1 - \beta)$, and then does the experiment.

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 and used p as a tool to make a decision that maintains long-term type I error rates at alpha 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 $p = 0.049$.

6.4.4   NHST

Modern researchers interpret p using a combination of Fisher and Neyman-Pearson concepts in what has become known as Null Hypothesis Significance Testing (NHST). Similar to Neyman-Pearson, a p-value is compared to α but similar to Fisher, many researchers, and many textbooks and statistics software (including base R) trichtomize a statistically significant p into "significance levels" (three asterisks for $p < 0.001$, two asterisks for $0.001 < p < 0.01$, and one asterisk for $0.01 < p < 0.05$) but many researchers also casuallly partition non-significant p values into "marginally signifiant" (or similar) and "not significant".

6.4.5   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.

6.4.5.1   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 probablity 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 probablity 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 "rigrously probe" the preferred hypothesis.

6.4.5.2   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.

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.
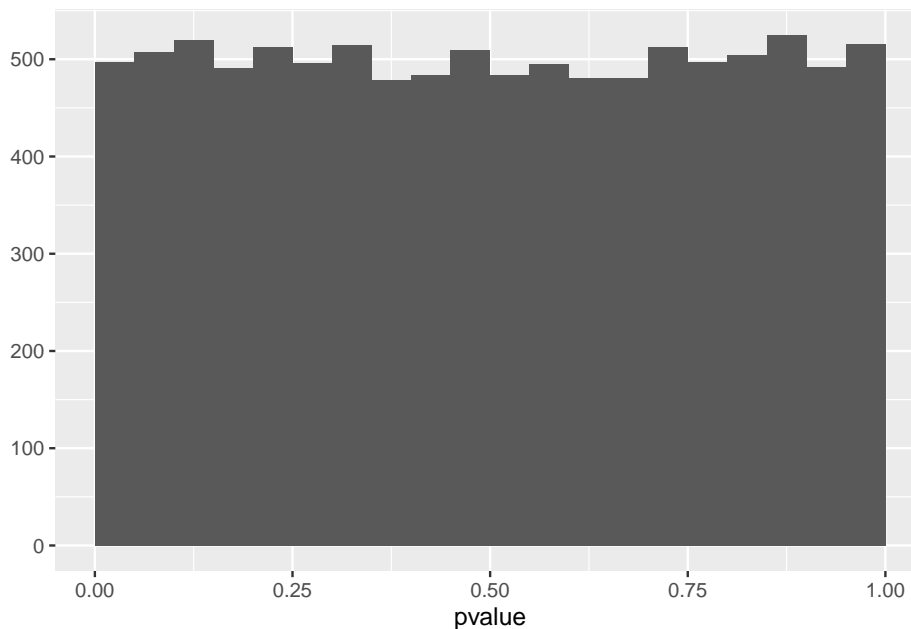
The incredible inconsistency of the p-value

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.

6.4.5.3 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 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 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
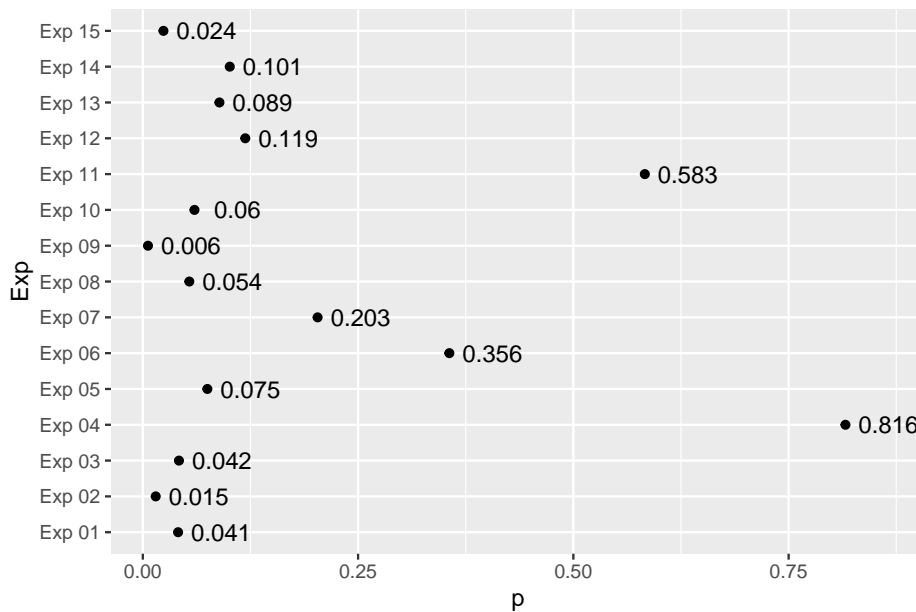
Figure 6.3: Variability of p-values when the power is 0.4

4. power is 25%. This means we expect to find $p \le 0.05$ 100 times (0.25×400) when the null is false
5. We have made $30 + 100 = 130$ "discoveries" (all experiments with $p \le 0.05$), 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 $p \le 0.05$ 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%).

6.4.5.4 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
```

6.4.5.5   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 capacity of a model because the p-value is a function of the signal, noise (unmodeled error), and *sample size* while predictive capacity is a function of just the signal:noise ratio. If the signal:noise ratio is tiny, the predictive capacity 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
```

6.4.5.5.1 What the p-value does not mean

1. p is not the probability of the null being true. More formally, this probability is $P(null|data)$ 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, H0 or H1.
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.

6.4.6 Recommendations

If you are working on basic science research simply report the exact p-value, along with a CI. 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. If p is smaller than say 0.001, then this is pretty good evidence that the data is not a fluke of sampling.

But 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.

If you are working on quality control then a p value is a useful tool, but is only relevant compared to a decision rule with well-reasoned values of α and β – exact values of p are not very meaningful.

6.5   Problems

Problem 1 – simulate the distribution of p under the null. There are many ways to do this but a straightforard 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 $p \le 0.05$. 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?

Problem 3 – write a script for a permutation test of the vitamin E and vitamin C levels of the vole data. Compare this to the t-test.

Problem 4 – grad students only. Simulate the false discovery rate. Explore the parameters: the frequency of true nulls and the power.

Chapter 7

Creating Fake Data

Fake data are generated by sampling from one of R's random sampling functions. These functions sample from different distributions including

1. uniform – function `runif(n, min=0, max=1)`, which samples n continous values between `min` and `max`.
2. normal (Gaussian) – function `rnorm(n, mean=0, sd=1)`, which samples n continous values from a distribution with the specified mean and standard deviation. The default is the "standard" normal distribution.
3. poisson – function `rpois(n, lambda)`, which samples `n` counts from a distribution with mean and variance equal to `lambda`.
4. negative binomial – `rnegbin(n, mu=n, theta)`, which samples `n` counts with mean `mu` and variance `mu + mu^2/theta`.

7.0.1   Continuous X (fake observational data)

A very simple simulation of observational design (the X are not at "controlled" levels)

```
n <- 25
# the paramters
beta_0 <- 25 # the true intercept
beta_1 <- 3.4 # the true slope
sigma <- 2 # the true standard deviation

x <- rnorm(n)
y <- beta_0 + beta_1*x + rnorm(n, sd=sigma)
qplot(x, y)
```

97
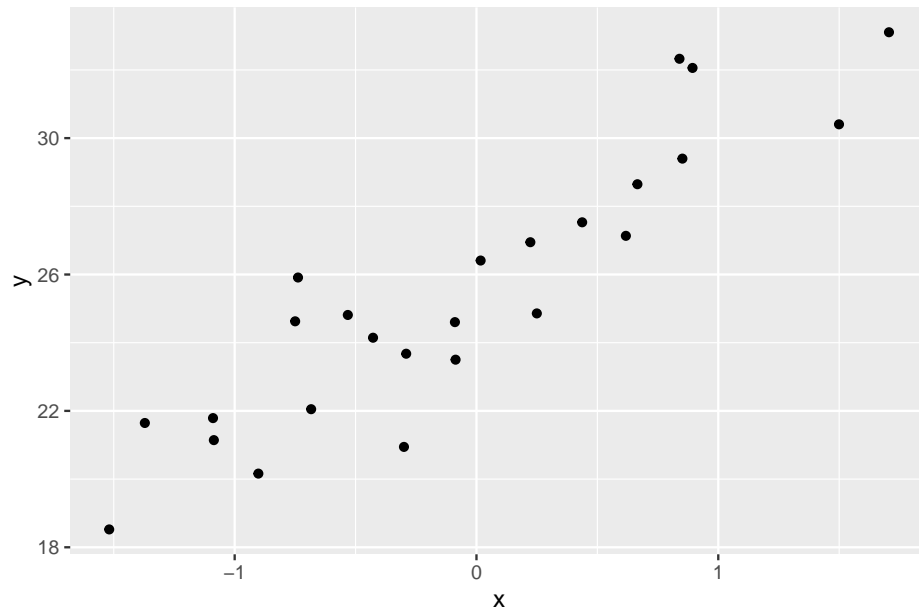
How well does a model fit to the data recover the true parameters?

```
fit <- lm(y ~ x)
knitr::kable(coefficients(summary(fit)), digits=c(1, 2, 1, 4))
```

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

25.8

0.34

75.8

0

x

4.2

0.40

10.4

0

The coefficient of x is the "Estimate". How close is the estimate? Run the simulation several times to look at the variation in the estimate – this will give you a sense of the uncertainty. Increase n and explore this uncertainty. Increase all the way up to $n = 10^5$. Commenting out the qplot line will make this exploration easier.

7.0.2 Categorical X (fake experimental data)

Similar to above but the X are at controlled levels and so this simulates an experimental design

```r
n <- 5 # the sample size per treatment level

fake_data <- data.table(Treatment=rep(c("control", "treated"), each=n))
beta_0 <- 10.5 # mean of untreated
beta_1 <- 2.1 # difference in means (treated - untreated)
sigma <- 3 # the error standard deviation
# the Y variable ("Response") is a function of treatment. We use some matrix
# algebra to get this done.
# Turn the Treatment assignment into a model matrix. Take a peak at X!
X <- model.matrix(~ Treatment, fake_data)
# to make the math easier the coefficients are collected into a vector
beta <- c(beta_0, beta_1)
# you will see the formula Y=Xb many times. Here it is coded in R
fake_data[, Response:=X%*%beta + rnorm(n, sd=sigma)]
# plot it with a strip chart (often called a "dot plot")
ggstripchart(data=fake_data, x="Treatment", y="Response", add = c("mean_se"))
```
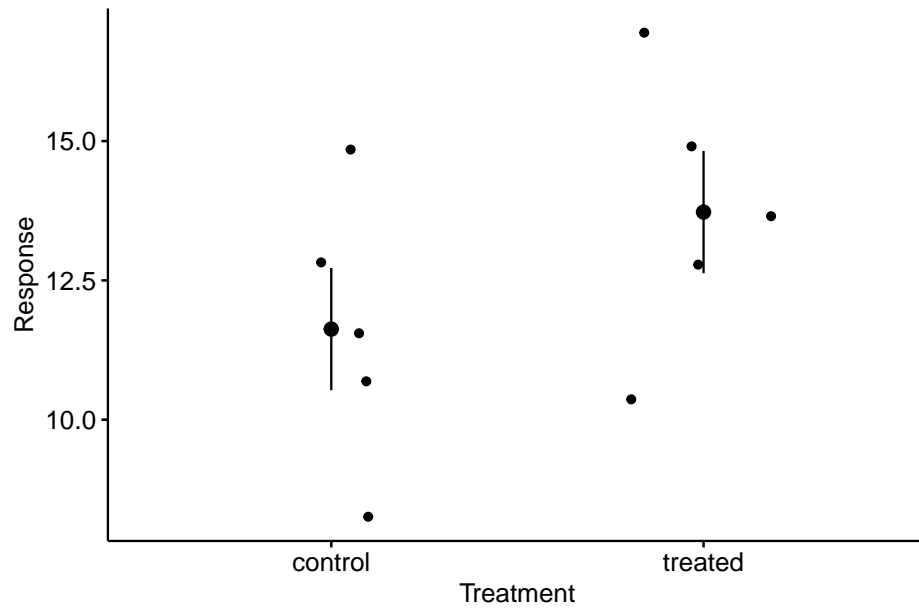
```
# fit using base R linear model function
fit <- lm(Response ~ Treatment, data=fake_data)
# display a pretty table of the coefficients
knitr::kable(coefficients(summary(fit)), digits=3)
```

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

11.626

1.097

10.601

0.000

Treatmenttreated

2.100

1.551

1.354

0.213

Check that the intercept is close to beta_0 and the coefficient for Treatment is close to beta_1. This coefficient is the difference in means between the treatment levels. It is the simulated effect. Again, change n. Good values are $n = 20$ and $n = 100$. Again, comment out the plot line to make exploration more efficient.

7.0.3 Correlated X (fake observational data)

7.0.3.1 Generating correlated X variables

It's useful to think about how correlated data are generated because often we want to generate fake_data with an expected correlation. Let's say we want to generate two X variables that have an expected correlation of 0.6. To generate this, we take advantage of the fact that two variables, X1 and X2, are correlated if they share a "common cause" – a variable Z that effects (or "causes") both X1 and X2. If the expected variances of X1, X2, and Z are all 1, then the expected correlation between X1 and X2 is the product of the causal effect from Z to each X. The easiest way to implement this is to simply make the effect from Z to both X equal to $\sqrt{(0.6)}$.

```r
n <- 10^3
z <- rnorm(n) # the common cause, with sigma = 1
rho <- 0.6 # the true correlation between X1 and X2
beta_z <- sqrt(rho) # the easiest way to get effects of z on X1 and X2 that generates rho
sigma_x <- sqrt(1 - rho) # we will make the variance of X1 and X2 = 1, so the "explained" varianc
x1 <- beta_z*z + rnorm(n, sd=sigma_x)
x2 <- beta_z*z + rnorm(n, sd=sigma_x)
# check
cov(data.frame(X1=x1, X2=x2)) # is the diagonal close to 1, 1?
```

```
##           X1        X2
## X1 1.0523598 0.5978408
## X2 0.5978408 0.9713292
```

```r
cor(x1, x2) # is the value close to rho?
```

```
## [1] 0.5913168
```

```r
qplot(x1, x2)
```
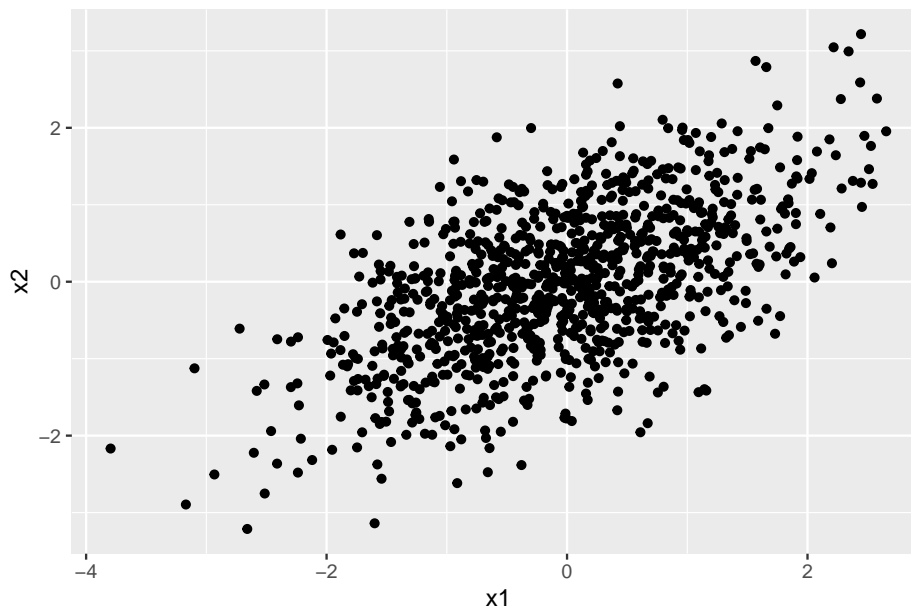
Now create fake Y that is a function of both X_1 and X_2. Create "standardized" fake data, where $\sigma_Y = 1$.

```r
beta_0 <- 3.2
beta_1 <- 0.7
beta_2 <- -0.3
explained_sigma <- beta_1^2 + beta_2^2 + 2*beta_1*beta_2*rho # Wright's rules! Compare
sigma_Y.X <- sqrt(1 - explained_sigma) # sqrt unexplained variance
y <- beta_0 + beta_1*x1 + beta_2*x2 + rnorm(n, sd=sigma_Y.X)

# check
var(y) # should be close to 1 as n gets bigger
```

```
## [1] 0.9791502
```

```r
# check parameters
coef(summary(lm(y ~ x1 + x2))) # should be near parameters as n gets bigger
```

```
##               Estimate Std. Error    t value      Pr(>|t|)
## (Intercept)  3.1887712 0.02536483 125.716255 0.000000e+00
## x1           0.6885565 0.03064195  22.471043 8.422461e-91
## x2          -0.3037457 0.03189446  -9.523465 1.223898e-20
```

Note that the variance of Y is the variance of the explained part due to X1 and X2 and the unexplained part and if the expected variance of $Y = 1$ then this sets an upper limit for the explained part. This means that

$$\beta_1^2 + \beta_2^2 + 2\beta_1\beta_2\rho < 1 \tag{7.1}$$

which means the magnitude of β_1 and β_2 should generally be less than 1.

7.0.3.2 Creating mulitple X variables using the package mvtnorm

The package mvtnorm provides a function to generate multivariate data (multiple columns) with a specified vector of means (the means of each column) and covariance matrix among the means.

```r
rcov1 <- function(p){
  # p is the number of columns or number of variables
  pp <- p*(p-1)/2 # number of elements in lower tri
  max_r <- 0.7
  r <- rexp(pp)
  r <- r*max_r/max(r)

  # create correlation matrix
  R <- matrix(1, nrow=p, ncol=p)
  R[lower.tri(R, diag=FALSE)] <- r
  R <- t(R)
  R[lower.tri(R, diag=FALSE)] <- r

  # convert to covariance matrix
  L <- diag(sqrt(rexp(p))) # standard deviations
  S <- L%*%R%*%L

  # check -- these should be the same
  # R
  # cov2cor(S)
  return(S)
}
```

Now let's use mvtnorm to generate fake correlated X

```r
p <- 5 # number of X variables
S <- rcov1(p)

# make the fake X
n <- 10^5
mu <- runif(p, min=10, max=100) # vecctor of p means
X <- rmvnorm(n, mean=mu, sigma=S)

# how close? (check the cor as this is easier to scan)
cov2cor(S)
```

```
##           [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 1.0000000 0.15511020 0.11863165 0.48788949 0.14545246
## [2,] 0.1551102 1.00000000 0.70000000 0.03514487 0.04345391
```

```
## [3,] 0.1186317 0.70000000 1.00000000 0.05346376 0.54083807
## [4,] 0.4878895 0.03514487 0.05346376 1.00000000 0.44504979
## [5,] 0.1454525 0.04345391 0.54083807 0.44504979 1.00000000
```

```
cor(X)
```

```
##               [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 1.0000000 0.15618693 0.12146387 0.48987841 0.14569745
## [2,] 0.1561869 1.00000000 0.69709529 0.03879653 0.03961509
## [3,] 0.1214639 0.69709529 1.00000000 0.05598629 0.54138395
## [4,] 0.4898784 0.03879653 0.05598629 1.00000000 0.44214983
## [5,] 0.1456974 0.03961509 0.54138395 0.44214983 1.00000000
```

7.0.3.3   The rcov1 algorithm is naive

A problem with generating a fake covariance matrix as above is that it is likely to be **singular** as p get's bigger. A singular covariance matrix is one where there are fewer orthogonal axes of variation then there are variables. Imagine a multidimensional scatterplot of a data set with the fake covariance matrix. If we zoom around this multidimensional space, we will come across a "view" in which all the points are compressed along a single line – that is there is no variation on the axis orthogonal to this line of points. This is bad, as it means we cannot fit a linear model using least squares (because the inverse of the covariance matrix doesn't exist).

Let's explore this. If the a covariance matrix is singular, then at least one eigenvalue of the matrix is negative (eigenvalues is a multivariate term beyond the scope of this text but, effectively, these are the variances of the orthogonal axes referred to above). Here I compute the frequency of covariance matrices with at least one negative eigenvalue as p increases

```
niter <- 1000
p_vec <- 3:10
counts <- numeric(length(p_vec))
i <- 0
for(p in 3:10){
  i <- i+1
  for(iter in 1:niter){
    counts[i] <- ifelse(eigen(rcov1(p))$values[p] < 0, counts[i]+1, counts[i])
  }
}
data.table(p=p_vec, freq=counts/niter)
```

```
##     p  freq
## 1:  3 0.000
## 2:  4 0.025
## 3:  5 0.066
```

```
## 4:   6 0.135
## 5:   7 0.190
## 6:   8 0.225
## 7:   9 0.275
## 8: 10 0.368
```

7.0.3.4 Generating multiple columns of X variables with a non-singular covariance matrix

This section uses some ideas from matrix algebra. The goal is to create a $n \times p$ matrix of X variables that have some random covariance structure that is full-rank (not singular, or no negative eigenvalues). The algorithm starts with a $p \times p$ random eigenvector matrix $\mathbf{E}$ and a $p \times p$ random eigenvalue matrix $\mathbf{L}$ and then computes the random covariance matrix using $\mathbf{ELE}^\top$

1. Generate a random $p \times p$ random eigenvector matrix from a covariance matrix of $p \times p$ matrix of random normal variables.

```r
fake.eigenvectors <- function(p){
    a <- matrix(rnorm(p*p), p, p) # only orthogonal if p is infinity so need to orthogonalize it
    a <- t(a)%*%a # this is the sum-of-squares-and-cross-product-matrix
    E <- eigen(a)$vectors # decompose to truly orthogonal columns
    return(E)
}
```

2. Generate p random eigenvalues in descending order and that sum to 1. There are several ways to create this sequence. Here are two:

```r
# force the eigenvalues to descend at a constant rate
fake.eigenvalues <- function(p, m=p, start=1, rate=1){
  # m is the number of positive eigenvalues
  # start and rate control the decline in the eigenvalue
  s <- start/seq(1:m)^rate
  s <- c(s, rep(0, p-m)) # add zero eigenvalues
  L <- diag(s/sum(s)*m) # rescale so that sum(s)=m and put into matrix,
  # which would occur if all the traits are variance standardized
  return(L)
}

# random descent
fake.eigenvalues.exp <- function(p, m=p, rate=1){
  # exponential distribution of eigenvalues
  # m is the number of positive eigenvalues
  # start and rate control the decline in the eigenvalue
  s <- rexp(m, rate)
  s <- s[order(s, decreasing=TRUE)] # re-order into descending order
  s <- c(s, rep(0, p-m)) # add zero eigenvalues
```

```
  L <- diag(s/sum(s)*m) # rescale so that sum(s)=m and put into matrix,
  # which would occur if all the traits are variance standardized
  return(L)
}
```

3. Generate the random covariance matrix

```
fake.cov.matrix <- function(p){
    # p is the size of the matrix (number of cols and rows)
    E <- fake.eigenvectors(p)
    L <- diag(fake.eigenvalues(p))
    S <- E%*%L%*%t(E)
    return(S)
}
```

4. Generate the random X variables using $\mathbf{X} = \mathbf{X}'(\mathbf{E}\sqrt{\mathbf{L}})^\top$

```
# two functions to compute the random data
fake.X <- function(n,p,E,L){
  # n is number of observations
  # p is number of variables
  X <- matrix(rnorm(n*p),nrow=n,ncol=p) %*% t(E%*%sqrt(L))
    return(X)
}
```

An example

```
n <- 10^5
p <- 5
E <- fake.eigenvectors(p)
L <- fake.eigenvalues(p, start=1, rate=1)
X <- fake.X(n, p, E, L)
colnames(X) <- paste0("X", 1:p)
E
```

```
##              [,1]        [,2]        [,3]        [,4]        [,5]
## [1,]   0.4762049  0.76679336  0.02189702 -0.37886958 -0.20306443
## [2,]  -0.3226816  0.26514622 -0.21147219  0.50680583 -0.72387940
## [3,]   0.6343127 -0.07414265  0.40381446  0.65434666  0.03024185
## [4,]   0.2450305 -0.53523178  0.22900376 -0.40906173 -0.65856872
## [5,]  -0.4546568  0.22305882  0.85982044 -0.06406746 -0.01166667
```

```
E <- eigen(cov(X))$vectors
scores <- X%*%E
colnames(scores) <- paste0("pc", 1:p)
cor(cbind(X, scores))[1:p, (p+1):(p*2)]
```

```
##           pc1        pc2         pc3        pc4          pc5
## X1  0.6381822  0.71536319  0.009665104  0.2581645 -0.119317350
```

```
## X2 -0.5701649  0.32632936 -0.213505507 -0.4289443 -0.582102493
## X3  0.8417769 -0.06005797  0.312645510 -0.4358132  0.011149679
## X4  0.4130194 -0.64271492  0.232581875  0.3517727 -0.488358607
## X5 -0.6579685  0.23893819  0.712352851  0.0501983 -0.004418134
```

Part III: Introduction to Linear Models

Chapter 8

A linear model with a single, continuous X

8.1 A linear model with a single, continuous X is classical "regression"

To introduce modeling with a single continuous X variable, I'll use data from

1. Source: Dryad Digital Repository. https://doi.org/10.5061/dryad.b3h4q
2. File: "FCM data dryad.csv"

The data are from **?**, who showed that North American red squirrel (*Tamiasciurus hudsonicus*) mothers from Yukon, Alaska produce faster growing pups in years with increased squirrel density. Remarkably, they even showed that perceived (but not actual) density results in faster growing pups. To begin to investigate how pregnant mothers control the future growth rate of pups, the researchers measured the relationship between local squirrel density and the amount of fecal cortisol metabolites from pregnant mothers. Cortisol is a hormone that is secreted as part of stress response. The researchers were interested in cortisol because it had previously been shownt that, in mammals, blood cortisol levels in pregnant mothers have numerous effects on offspring long past birth. If increased squirrel density causes increased blood cortisol levels then we would expect to find a positive relationship between *Density* and

Figure 8.1 is a **scatterplot** of the data with the amount of cortisol metabolites in the feces on the Y axis and local squirrel density on the X axis. The line through the data is a graphical representation of a linear model fit to the data and the gray cloud around the line is a graphical representation of the uncertainty in the model. The researchers wanted to model the "effect" of squirrel density on the
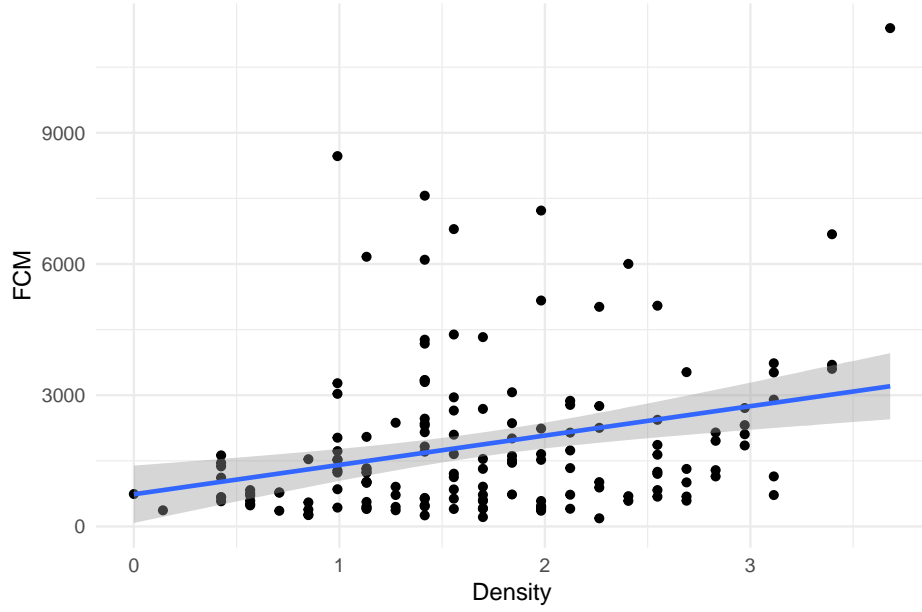
Figure 8.1: A scatterplot of Fecal cortisol matabolites and squirrel density.

amount of cortisol metabolites in the feces of the pregnant mothers. Graphically, this effect is the slope of the line in Figure 8.1.

The model fit to the data is

$$FCM_i = \beta_0 + \beta_1 Density_i + \varepsilon_i \tag{8.1}$$

which contains both the linear predictor and the error. For inference, for example, computing standard errors of the coefficients, We need to model the error. Here, we use the simplest model of error which is "IID $N(0, \sigma)$". This means, the modeled error is

1. Independent – individual error values are independent of other values.
2. Identical – individual error can be thought of as a sample from a single **random distribution** (the same for each individual value). For this model, this distribution is
3. $N(0, \sigma)$ – the modeled distribution is "Normal" or "Gaussian", with a mean of zero and a standard deviation of σ.

The predictor part of the model is

$$\mathrm{E}[FCM|Density] = \beta_0 + \beta_1 Density \tag{8.2}$$

In words, model (8.2) reads "the expected value of FCM conditional on density is beta-knot plus beta-one times density". An **expected value** is a long run average – if we were to sample lots and lots of red squirrel populations with $Density = x$ (where x is a specific value), we'd expect the average FCM across these samples to be $\beta_0 + \beta_1 x$.

> Let's unpack this. $E[Y]$ is the **expectation** or **expected value** of Y. An expectation is the long-run average of Y if we were to run an experiment or re-sample a population many times. The sample mean of Y is an estimate of $E[Y]$. $E[Y|X]$ is a conditional expectation of Y – it is the expectation given additional conditions. Using the red squirrel example, these conditions are a specific value of $Density$. If FCM is linearly related to $Density$ (the right-hand side of equation (8.2)) then the expected value of FCM given a local density of 2.8 squirrels differs from the expected value of FCM given a local density of 1.4 squirrels (the units of $Density$ are squirrels per 150 meter radius of the individual female's midden).

In model (8.2), there is a single X variable (FCM). While the X variables are often called the "dependent" variables, in this model FCM does not "depend" on the independent variable $Density$ in any causal sense – meaning if I were to intervene and set $Density$ to some value x, I would expect FCM to equal $\beta_0 + \beta_1 x$. Rather, FCM only "depends" on $Density$ in a probablistic sense – if $Density = x$ then the most probable value of FCM is $\beta_0 + \beta_1 x$. With some strong assumptions model (8.2) can be turned into a model of causal dependency, which is the focus of chapter xxx.

β_0 and β_1 are the **parameters** of model (8.2). Specifically β_0 is the model **intercept** and β_1 is the modeled **effect** of $Density$. Again, the effect (β_1) has a probabilistic, and not causal, interpretation. This interpretation is

$$\beta_1 = \mathrm{E}[FCM|Density = x + 1] - \mathrm{E}[FCM|Density = x] \qquad (8.3)$$

Or, in words, "beta-1 is the expected value of FCM when density equals x + 1 minus the expected value of FCM when the density equals x." β_1 is simply the difference in expected values given a one unit difference in $Density$. A very short way to state this is "β_1 is a difference in conditional means".

8.1.1 Using a linear model to estimate explanatory effects

The goal of the statistical model here is to estimate β_1 – the probablistic effect of $Density$ on FCM. This estimate, and a measure of the uncertainty of this estimate, are in the table of coefficients of the fit model

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

736.0

331.9

2.2

0.0281

Density

671.1

178.9

3.8

0.0002

where the entries in the column "Estimate" are estimates of the parameters β_0 and β_1 in model (8.2). The entries in the column "Std. Error" are the standard errors (SE) of the estimates, which are measures of the uncertainty of the estimates.

The parameter estimates in the table above are the coefficients of the fitted model

$$FCM_i = b_0 + b_1 Density_i + e_i \qquad (8.4)$$

where the subscript i refers to the ith individual. The coefficients b_0 and b_1 are the y-intercept and the slope of the line in Figure 8.1. The coefficient for *Density* (b_1) is 671.1, and (given the definition of the parameter β_1 in equation (8.3)) we expect squirrel mothers with a local density of 2 squirrels within a 150 m radius of her midden to average 671.1 more units of FCM (ng of fecal cortical metabolites per gram dry food) than mother squirrels with a local density of only 1 squirrel within a 150 m radius of her midden.

8.1.1.1 Probabilistic vs. causal conditioning

Remember that this coefficient is estimating a probabilistic parameter. Consequently, the coefficient b_1 is simply a descriptor of a pattern of relationship between local density and fecal cortisol metabolites - no causal effect is implied. With the strong assumptions explained in chapter xxx, however, b_1 can estimate a causal effect.

8.1.2 Using a linear model for prediction

Model (8.4) gives the measured value of *FCM* for each squirrel. The equation includes the linear predictor ($b_0 + b_1 Density_i$) and the **residual** from the predictor (e_i). The predictor part is called "predictor" because it is the equation for predicting the value of an individual's *FCM* given that individual's value of *Density*:

$$\widehat{FCM} = b_0 + b_1 Density \tag{8.5}$$

where $\widehat{FCM}$ is read as "FCM hat" and is the **predicted value** or simply "prediction". Very often, we use the predictor part (equation (8.5)) to predict unknown or future values given different modeled inputs (the X).

8.1.3 Reporting results

The authors of the squirrel fcm data published a figure and table similar to fig. xxx and table above but used a slightly more complex linear model. Here is how the author's reported the results:

> Across 6 years (2006 to 2011), we found a positive relationship between local density and concentrations of fecal cortisol metabolites [FCM; $t_1 55 = 3.63$, P = 0.0002 (table S4 and Fig. 3A)].

I would advocate reporting the estimate and a confidence interval instead of t and p. For example "Across 6 years (2006 to 2011), the probabilistic effect of local density on fecal cortisol metabolites is 671.1 (95% CI: 317.7, 1024.5). If a p-value is report *in addition* to the effect and CI, always report the exact p-value, which emphasizes the continuous nature of evidence against the null, and not something like"$p < 0.05$", which artificially dichotomizes the evidence against the null.

8.2 Working in R

8.2.1 Exploring the bivariate relationship between **Y** and **X**

Questions

1. Import the "FCM data dryad.csv" data from the Dryad repository as the data.table `fcm`
2. How are different words in the column labels demarcated? Is this good practice?

Here we want to fit a model of `FCM.ng.g.dry` as a function of `Raw.Squirrel.Density`. The authors used prior knowledge to expect a positive relationship between these two variables. Use qplot to generate a scatterplot of *FCM* against *Density*

Questions

   3. Is there a trend? If so, does the trend look linear or non-linear?
   4. Does the residual variation (the deviation from the trend on the *Y* axis) look homogenous along the *X*-axis?
   5. Are there any obvious outliers?

8.2.2   Fitting the linear model

We will fit a linear model to the data using the `lm` function, which is very general and will be our workhorse throughout the class. The minimal input to the function is a model formula and the name of the data.frame (remember, a data.table is a data.frame). A formula is of the form `Y ~ X`. All of the output we assign to the object `fit`.

Let's fit the linear model to the data using density as the predictor

```
fit <- lm(FCM.ng.g.dry ~ Raw.Squirrel.Density, data=fcm)
```

R will look for the specified *Y* and *X* variables in the column names of `fcm`. If these are not found, R will return an error, for example

```
fit <- lm(FCM_ng_g_dry ~ Raw_Squirrel_Density, data=fcm)
```

will return the error "Error in eval(predvars, data, env) : object 'FCM_ng_g_dry' not found". This means your spelling and capitalization have to be exact!

8.2.3   Getting to know the linear model: the `summary` function

The `lm` function returns an `lm` object, which we've assigned to the name `fit`. `fit` 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 the works with many R objects.

```
summary(fit)
```

```
##
## Call:
## lm(formula = FCM.ng.g.dry ~ Raw.Squirrel.Density, data = fcm)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2107.5 -1108.3  -434.9   511.8  8186.8
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)            736.0      331.9   2.217 0.028078 *
## Raw.Squirrel.Density   671.1      178.9   3.752 0.000248 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1732 on 154 degrees of freedom
##   (7 observations deleted due to missingness)
## Multiple R-squared:  0.08374,    Adjusted R-squared:  0.07779
## F-statistic: 14.07 on 1 and 154 DF,  p-value: 0.0002484
```

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 (8.4)). Here, 735.9604344 is the intercept (b_0) and 671.1379749 is the effect of *Density* (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. Any t less than 2 is indicative of too much noise to say much about the signal. A t between 2 and 3 means the noise is large enough to suggest an effect. A t greater than 3 is pretty good evidence of an effect.
4. The column of values under "Pr(>|t|)" is the p-value, which is the exact probability associated with a particular t. What is the p-value a test of? The p-value tests the hypothesis "how probable are the data if the coefficient is zero?". Formally $P = \text{freq}(t' \geq t | H_o)$, where t' is the hypothetical t-value, t is the observed t-value, and H_o is the null hypothesis. We will return to p-values in Chapter xxx.

Signif. codes. I am surprised that base R returns this. These are useless because the concept of "levels of significance" is muddled, as will be discussed in Chapter xxx.

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 fit. 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**.

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.

Note that the p-value for the coefficient for Raw.Squirrel.Density is very small and we could conclude that the data are not consistant with a model of no slope. 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". What we want to know is not *if* there is a relationship between *FCM* and *Density*, which is imperfectly answered with a p-value, but *the sign and magnitude* of the relationship and the uncertainty in this estimate. For this, we don't need the p-value. Instead, we want to interpret the coefficient to its SE directly (for a quick-and-dirty interpretation) or the confidence interval of the effect (for a more formal interpretation). Please read this paragraph again. We will come back to it over and over.

8.2.4   display: An alternative to summary

Much of what we want to know about a model fit is returned by the `display` function from the `arm` package.

```
display(fit)
```

```
## lm(formula = FCM.ng.g.dry ~ Raw.Squirrel.Density, data = fcm)
##                      coef.est coef.se
## (Intercept)          735.96   331.94
## Raw.Squirrel.Density 671.14   178.90
## ---
## n = 156, k = 2
## residual sd = 1732.02, R-Squared = 0.08
```

The `display` function does not give a t-value or a p-value of the coefficients because the authors of the arm package do not think p-values are very valuable. We don't need a t because one can mentally compute the approximate ratio of

the coefficient to its SE and get a sense of the signal to noise, and that's all the authors of the display function think we need.

8.2.5 Confidence intervals

Confidence intervals for the coefficients of the model are obtained by

```r
confint(fit)
```

```
##                          2.5 %    97.5 %
## (Intercept)           80.21785 1391.703
## Raw.Squirrel.Density 317.73057 1024.545
```

`confint` returns by default the 95% confidence interval (CI) of all parameters. The most useful way of thinking about the meaning of a CI is

A confidence interval contains the range of parameter values that are consistent 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 "consistent" 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. Remember this.

8.2.6 How good is our model?

How well does variation in *Density* "explain" variation in *FCM*? The answer to this is in the R^2 value, which is given in `display(fit)` and in `summary(fit)` and accessed directly with

```
summary(fit)$r.squared
```

```
## [1] 0.08373756
```

R^2 is the fraction of the total variance of Y explained by the model, or more specifically, the linear predictor. It will vary from zero (the model explains nothing) to one (the model explains everything). If $R^2 = 0$ the response is completely unpredictable by the predictors. We can think of the values of the response as white noise or all error. This doesn't mean that the values are "not caused" or "random" or not predicted by some other variable. It only means the values are random with respect to the X variable(s) in the model. If $R^2 = 1$ we can *exactly* predict the response from the X variables in the model. So the bigger the R^2, the better the model in the sense that the response is more predicatable. **Super importantly**, "explains" is in a probabilistic and not causal sense. We will explore this concept much more in future worksheets.

8.2.7 Model checking

`plot` is a very useful base R function for "model checking" or "model diagnostics" to see if our model fit is acceptable.

```
plot(fit)
```


lm(FCM.ng.g.dry ~ Raw.Squirrel.Density)

Normal Q–Q

Standardized residuals

Theoretical Quantiles
lm(FCM.ng.g.dry ~ Raw.Squirrel.Density)

Scale–Location

√|Standardized residuals|

Fitted values
lm(FCM.ng.g.dry ~ Raw.Squirrel.Density)

Residuals vs Leverage

lm(FCM.ng.g.dry ~ Raw.Squirrel.Density)
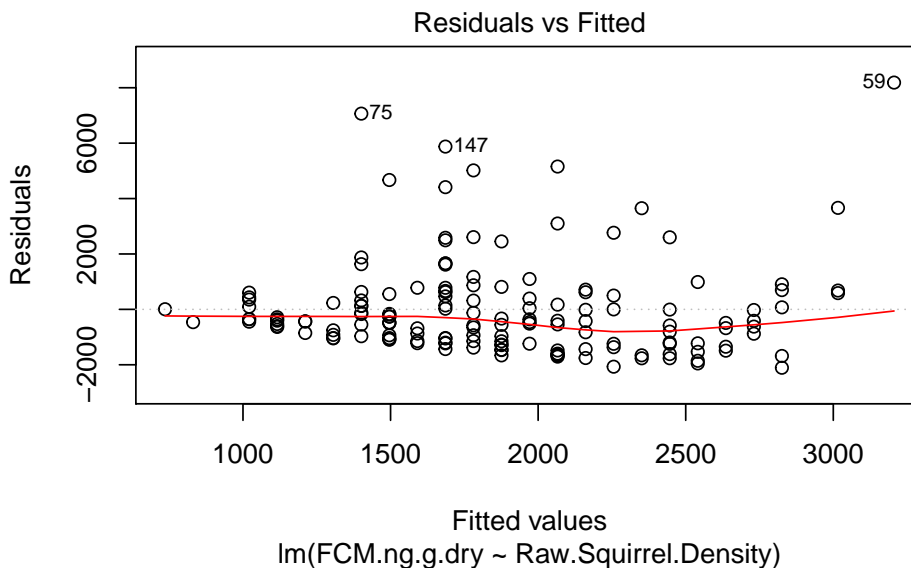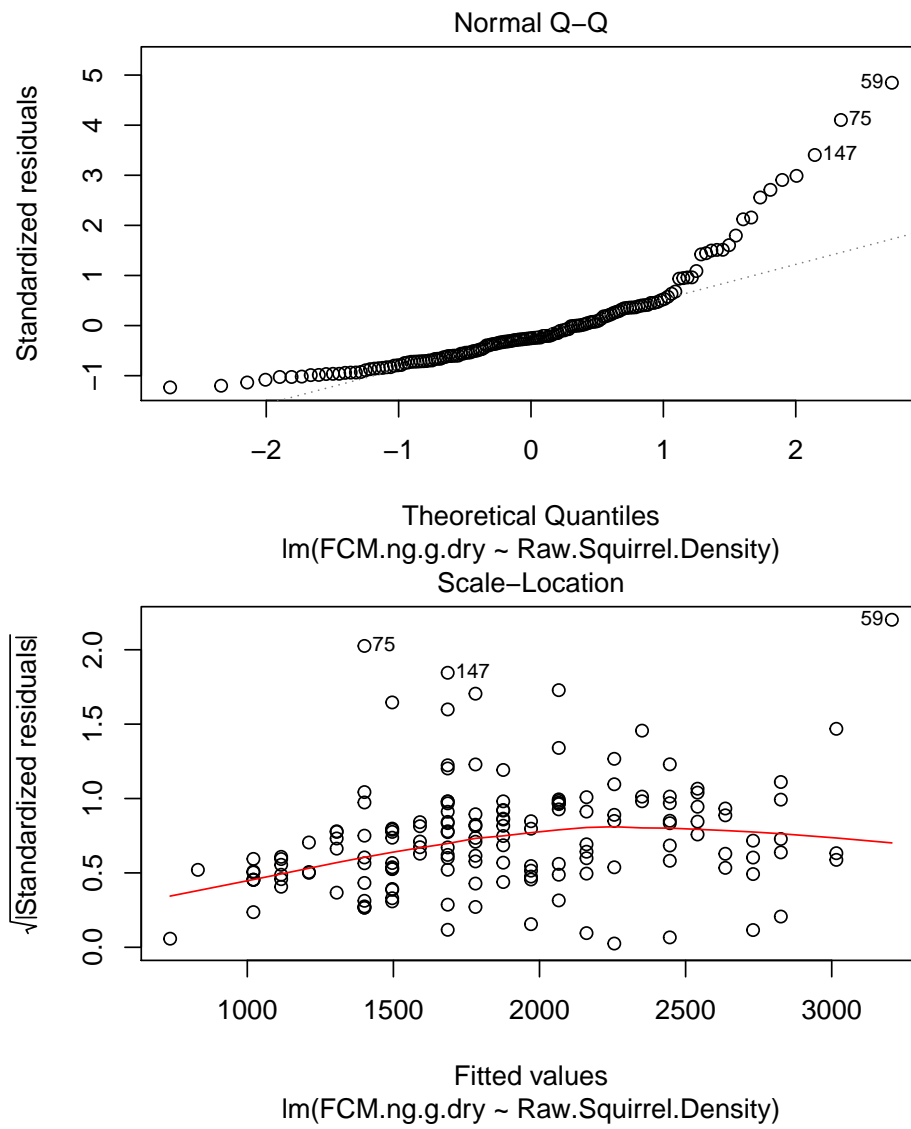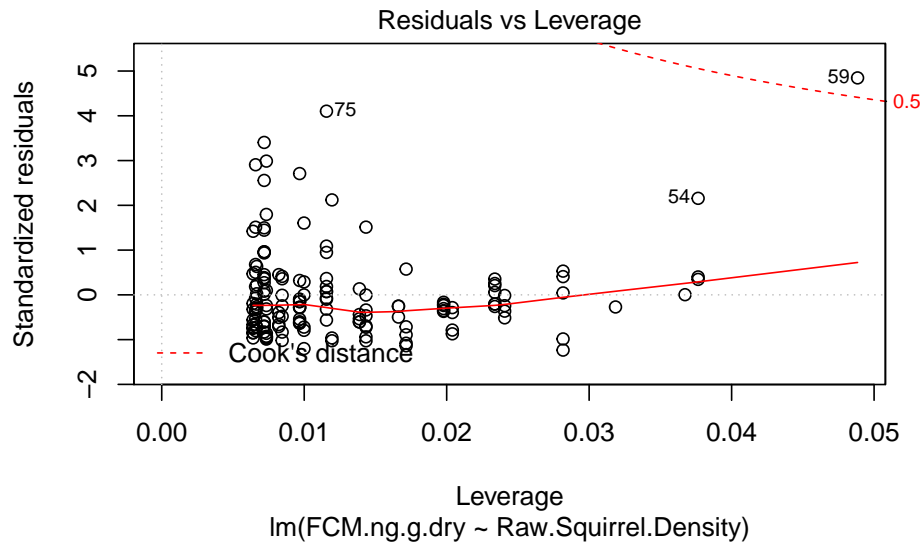
Compare the four diagnostic plots using the guidelines from here http://data.library.virginia.edu/diagnostic-plots/

Questions

6. Look at the plots you just made. What is a residual? What is a fitted value?

8.2.8 exploring a lm object

`fit` contains much information but simply typing `fit` into the console gives us only the model and the coefficients. `names` is a super important R function. It gives us the names of all the parts of some R object. `fit` is an lm object. `names(fit)` gives us all the parts contained in an lm object.

```
names(fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "na.action"     "xlevels"       "call"          "terms"
## [13] "model"
```

You can see any of these parts using the dollar sign

Questions

7. What does `fit$residuals` return? Answer using equation (8.4)

8. What does `fit$fitted.values` return? Answer using equation @ref(eq:fcmi

You can use qplot to make a plot similar to the first plot of `plot(fit)`

```
qplot(fit$fitted.values, fit$residuals, geom=c('point', 'smooth'))
```

`geom_smooth()` using method = 'loess' and formula 'y ~ x'
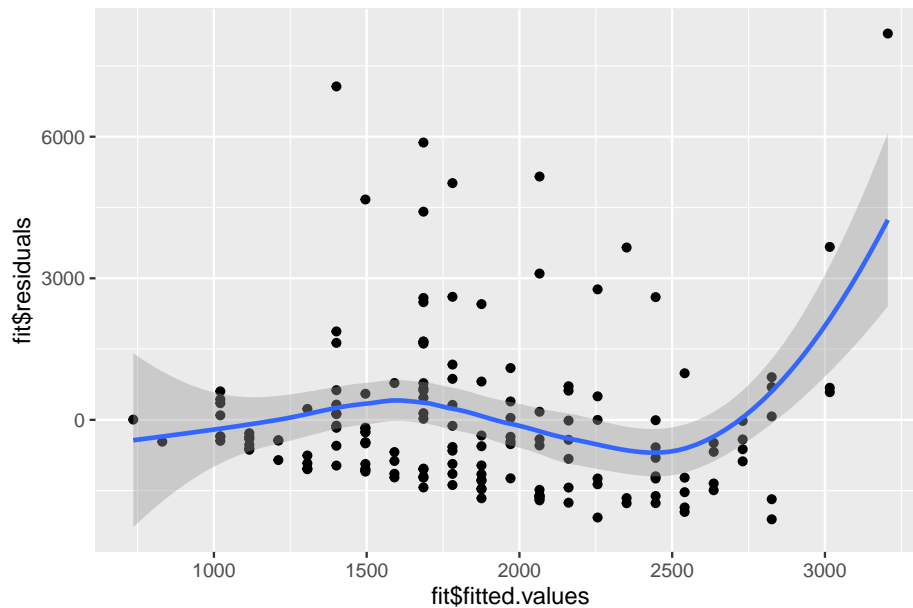
8.3 Problems

1. Using the chick data from Chapter 3. Compare the effects of nest_temperature_above_ambient on day13_mass by fitting two separate linear models 1) one using only the control group and one using the treated group. The grouping variable is playback_treatment. These models were plotted in Chapter 3 so `lm` will return the linear model behind these plots.

Report the results using the two effect estimates and a 95% confidence interval (we will learn in a later chapter a more sophisticated way of comparing the effects between the groups)

file name: "allDatasetsMarietteBuchanan2016.xls"

source: https://datadryad.org//handle/10255/dryad.122315

2. (Grad students only) – find a dataset using Dryad that has data that can be fit by a simple linear model with a single continuous X (its okay if the authors fit the data with a more complex model). Fit the data and report the results with a plot and text.

Chapter 9

A linear model with a single, categorical X

9.1 A linear model with a single, categorical X estimates the effects of X on the response.

To introduce modeling with a single, categorical X variable, I'll use the Vole data from Chapter 2. Normal cellular metabolism creates reactive oxygen species (ROS) that can disrupt cell function and potentially cause cell damage. Anti-oxidants are molecules that bind ROS, inhibiting their ability to disrupt cell activity. A working hypothesis for many years is that supplemental anti-oxidants should improve cell function and, scaling up, whole-animal function (such as lifespan). The vole data explores this with supplemental Vitamins C and E, which are anti-oxidants, in the diet of the short-tailed field vole (*Microtus agrestis*).

The goal of the study is to measure the effect of anti-oxidants on lifespan. The researchers randomly assigned the voles to one of thre treatment levels: "control", "vitamin E", and "vitamin C". The variable *treatment*, is a single, categorical X variable. Categorical variables are often called **factors** and the treatment levels are often called **factor levels**. "Levels" is a strange usage of this word; a less formal name for levels is "groups". There are no units to a categorical X variable (even though a certain amount of each anti-oxidant was supplemented). The response (Y) is *lifespan* measured in days.

The linear model with a categorical X variable with three levels is not immediately obvious, and so I don't present the model until after showing the table of model coefficients. The verbal model is

$$lifespan \sim treatment \qquad (9.1)$$

which can be read as "lifespan as a function of treatment".

9.1.1 Table of model coefficients

The *table of coefficients* from a linear model fit to some data is critically important for understanding a linear model and interpreting results. Read this section carefully. The coefficient table for a linear model fit to the vole data is

Coefficient table of fit linear model of vole data.

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

503.4

27.4

18.4

0.000

treatmentvitamin_C

-115.1

54.5

-2.1

0.037

treatmentvitamin_E

-89.9

52.5

-1.7

0.090

The table has three values in the column "Estimate". The first estimate, that for "(intercept)" is the mean response in the reference level. Here, the reference level is the "control" group. The second estimate, that for "treatmentvitamin_C" is the difference between the mean of the vitamin C group and the the mean of the reference (control) group. The *direction* of this difference is important; it is $\bar{Y}_{vitamin\_c} - \bar{Y}_{control}$, that is the non-reference level minus the reference level. The third estimate, that for "treatmentvitamin_E" is just like the second
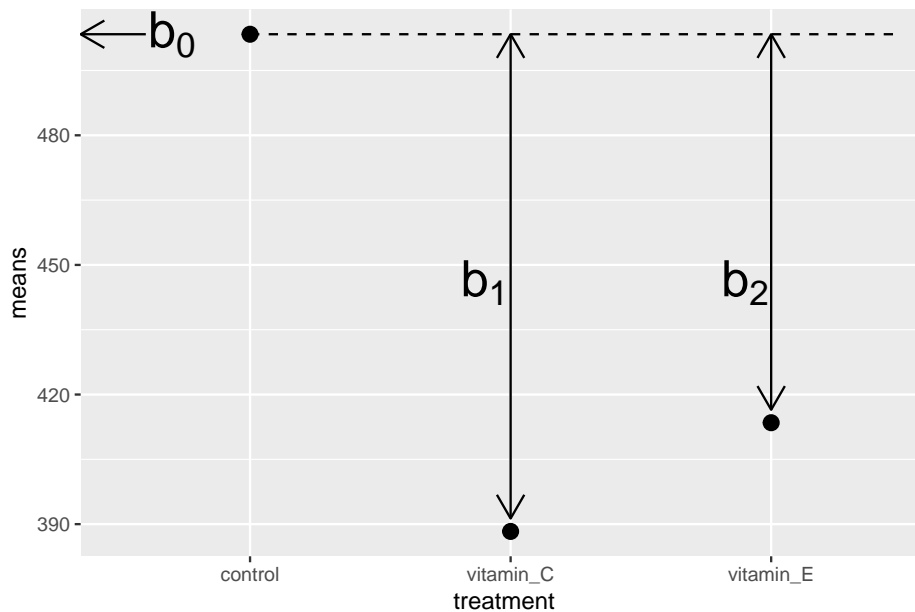
Figure 9.1: What the coefficients of a linear model with a single categorical X mean. The means of the three treatment levels for the vole data are shown with the filled circles. The length of the double-headed arrows are differences in means. The intercept (b_0) is the mean of the reference treatment level. The coefficients $(b_1$ and $b_2)$ are the differences between the treatment level's mean and the reference mean. As with a linear model with a continuous X, the coefficients are effects.

estimate, except for the vitamin E group. That is, it is $\bar{Y}_{vitamin\_e} - \bar{Y}_{control}$. The 2nd and 3rd values in the "Estimate" columns are the "effects" in the model. These effects are "what happens" when we add a treatment, such as vitamin E supplementation. When we add the vitamin E supplement, we find the lifespan changes by -89.9 days, relative to the control.

So typically with categorical X, when we speak of an *effect* we mean a difference in means, or a **contrast**.

9.1.2 The linear model

We can see an immediate difference between the coefficient table for a linear model fit to a single, categorical X and that for a single, continuous X. For the latter, there is a single coefficient for X. For the former, there is a coefficient for each level of the categorical X *except* the "reference" level.

The linear model for a single, categorical X with three factor levels is

$$Y = \beta_0 + \beta_1 X_{group2} + \beta_2 X_{group3} + \varepsilon \tag{9.2}$$

where *group*2 and *group*3 refer to the two non-reference groups.

For the vole data, "control" is the reference, so the model is

$$lifespan = \beta_0 + \beta_1 vitamin\_C + \beta_2 vitamin\_E + \varepsilon \tag{9.3}$$

The "estimates" in the coefficient table are the estimates of the parameters in this linear model. These estimates are the coefficients of the fit model,

$$lifespan_i = b_0 + b_1 vitamin\_C_i + b_2 vitamin\_E_i + e_i \tag{9.4}$$

Given the interpretation of the estimates above, b_0 is the mean of the control group, b_1 is the difference in means between the vitamin C and control groups, and b_2 is the difference in means between the vitamin E and control groups. These estimates and their meaning are illustrated in Figure 9.1. Take a while to memorize the bold-faced sentence above equation (9.4) and to understand this plot. Be able to "visualize" the meaning of the coefficients of a linear model in this way.

9.1.2.1   A linear model with a categorical X is a regression model with the treatment levels re-coded as numbers

Model (9.4) is a regression model. A regression model requires that the X variables be numeric, so how can this model be a regression model? What are the numeric values of *vitamin_C* and *vitamin_E*? The answer is very clever: *vitamin_C* and *vitamin_C* are **dummy variables** that contain a one, if response i is from that treatment level, and zero otherwise. This is called dummy coding or treatment coding. The `lm` function creates these dummy variables under the table, in something called the **model matrix**, which we'll cover in another chapter. You won't see these columns in your data. But if you did, it would look something like this

lifespan

treatment

vitamin_E

vitamin_C

621

control

0

0

865

control

0

0

583

vitamin_E

1

0

561

vitamin_E

1

0

315

vitamin_C

0

1

157

vitamin_C

0

1

There are alternative coding methods. Dummy coding is the default in R and it makes since when thinking about experimental data. Note that the method of coding can make a difference in an ANOVA table, and many published papers using R have published incorrect interpretations of ANOVA table outputs. This is both getting ahead of ourselves and somewhat moot, because I don't advocate publishing ANOVA tables.

9.1.2.2 Some math to convince you that the intercept of a linear model with a categorical X is the mean of the reference group *and* the intercept of a line. And some math to convince you that the coefficient of a dummy variable in a linear model with a categorial X is a difference in means *and* a slope.

The interecept of a model is the value of the model when all X-variables are set to zero. The X variables in the model (Equation (9.4)) are the dummy variables *vitamin_E* and *vitamin_C*. If we set vitamin_E$ and *vitamin_C* in Equation (9.4) to zero, the modeled (or expected) value reduces to

$$\mathrm{E}(lifespan|X_1 = 0, X_2 = 0) = b_0 \tag{9.5}$$

% Since both dummy variables are set to zero, we have modeled the expected value or mean of the control group.

The slope of a model is the difference in the modeled value given a one unit increase in X. If we increase the dummy variable *vitamin_E* from zero to one (that is, if we are modeling the expected value of the vitamin E group), we get

$$\mathrm{E}(lifespan|X_1 = 1, X_2 = 0) = b_0 + b_1 \tag{9.6}$$

which can be re-arranged to

$$b_1 = \mathrm{E}(lifespan|X_1 = 1, X_2 = 0) - b_0 \tag{9.7}$$

and since $\mathrm{E}(lifespan|X_1 = 0, X_2 = 0) = b_0$ then

$$b_1 = \mathrm{E}(lifespan|X_1 = 1, X_2 = 0) - \mathrm{E}(lifespan|X_1 = 0, X_2 = 0) \tag{9.8}$$

or, the coefficient of vitamin C is the difference in means between the vitamin C and control groups, which is also a slope since this is the expected difference given a one unit increase in *vitamin_C*.

9.1.3 Reporting results

What should be reported for the analyis of effects of anti-oxidant supplements on vole lifespan? Best practice includes reporting the raw data with a summary distribution and treatment effects with CIs. "Raw data" means the individual lifespans as a function of treatment level.
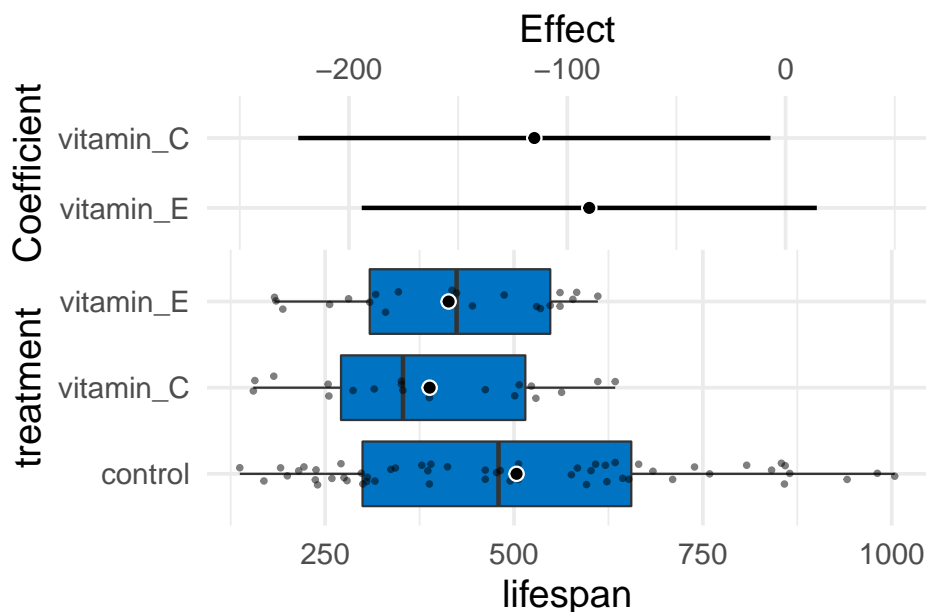
Figure 9.2: HarrellPlot of the raw data, distribution, and effects of the vole lifespan data.

9.1.3.1   Harrell Plot of the data

The raw data, the distributions within treatment level, and the effects (difference in means) of treatment can be combined into a single plot that I call a Harrell plot (Figure **??**). Notice that the x-axis and y axes are flipped so that $lifespan$ is on the x-axis. It is still the "response" or "Y" variable! The Harrell plot contains two parts

1. The bottom contains a **strip chart** (often called a "dot plot") of the raw response measures grouped by factor level. Superimposed over the strip chart is a **box plot** summarizing the distribution of each factor level. The line in the center of a box is the median $lifespan$ for that group, the left and right edges of the box are the 25% and 75% quantiles of $lifespan$ for that grop, and the lines extending to the left and right of the box are the "whiskers", which are the smallest and largest value within $1.5IQR$ (inter-quartile range, which is the interval bounded by box).
2. The top is a **forest plot** of the effects and the 95% CI of the effects. For categorical X, the effects could be model coefficients or treatment **contrasts**, which are differences in means between treatment levels. Model coefficients are a subset of possible treatment contrasts.

The Harrell plot above shows the effects as model coefficients, which (again!) are differences between the mean of the response in a specific treatment level

and the mean of the response in a reference level. Here the reference level is the control group.

9.1.3.2 In-text reporting

"The mean lifespan of cold-reared voles supplemented with vitamin E was 89.9 days shorter than the mean lifespan for the control group (95% CI: -194.1, 14.3). The mean lifespan of cold-reared voles supplmented with vitamin C was 115.1 days shorter than the mean lifespan for the control group (95% CI: -223.2, -6.9).

9.1.3.3 Correct interpretation of the Confidence Interval is key

Remember, that the CI contains the range of parameter values that are consistent with the data (in the sense that a t-test wouldn't reject the hypothesis test). This means that a true value at the low end or the high end of the CI is consistent with the data. Your technical report/manuscript should discuss the consequences of this. For example, A small, increase in lifespan is consistant with the Vitamin E but not Vitamin C supplementation, if we use the 95% CI as a pretty good range for inferring "consistent with". Both a 223 day and a 7 day decrease in lifespan are consistant with the Vitamin C effect. 223 days seems like a huge effect, especially for a short lived vole. 7 days is certainly a much smaller effect, but this doesn't mean that it doesn't have important ecological, behavioral, or fitness consequences.

9.2 Comparing the results of a linear model to classical hypothesis tests

9.2.1 t-tests are special cases of a linear model

There isn't "a" *t*-test but several flavors of *t*-test including

1. Student's *t*-test. The classical "two-sample" test for comparing the means between two groups
2. Welch's *t*-test. A modification of Student's test, which relaxes the assumption of equal variance between the groups.
3. paired *t*-test. A version of the test when values in the two groups are "paired", for example, measuring weight in ten mice before treatment (at "baseline"), measuring weight in the same ten mice after treatment, then comparing the mean post-treatment to mean pre-treatment weight.

All of these are special cases of the linear model. Welch's and paired t-test are swept within a linear model in later chapters. Here, I focus on Student's *t*-test.

First, let's review t-values, which were introduced in Chapter 5 on p-values. A reminder, a t-value is a ratio of signal to noise, where the signal is an estimate of some parameter and the noise is the standard error of the estimate. The parameter of interest here is the difference in means between treatment and control, so t is

$$t = \frac{\bar{y}_t - \bar{y}_c}{s_{\bar{y}_t - \bar{y}_c}} \tag{9.9}$$

Note that the numerator and denominator in equation (9.9) are in the coefficient table of a linear model with categorical X – the numerator is the estimate of the effect of a treatment and the denominator is the standard error of this estimate.

To explore these equalities, let's use data from

article 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.

data source https://datadryad.org/stash/dataset/doi:10.5061/dryad.6121hj7

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.

9.2.1.1 A student t-test is equivalant to the t-value and p-value in a coefficient table of a linear model *if* there are only two levels in the treatment factor

Here I compare pre-insulin infusion blood levels of free fatty acids (ffa) between obese and lean subjects at 12h. The data are in Table 2 and the response is the column "ffa_t_210_min_m_m". The assigment of lean or obese is in Table 1, which needs to be merged with Table 2 in order to subset the lean subjects.

Coefficient table from the linear model

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.417625 0.04433893 9.418923 1.950636e-07
## groupobese  0.105625 0.06270472 1.684482 1.142414e-01
```

The t-value and p-value of the effect of obesity on free-fatty acids *is* a t-test. The numerator of t is the difference in free-fatty acids between obese and lean subjects (the "Estimate" in the coefficient table). The denominator of t if the standard error of this estimate (The "Std. Error" in the coefficient table).

To confirm that that the the $t$and p-values of the effect of obesity on free-fatty acids *is* a t-test, let's compare the coefficient table to the output of a t-test.

```
##
##   Two Sample t-test
##
## data:  ffa_t_210_min_m_m by group
## t = -1.6845, df = 14, p-value = 0.1142
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.24011325  0.02886325
## sample estimates:
##   mean in group lean mean in group obese
##             0.417625             0.523250
```

The t-value in the t.test output is the same as the t-value of the effect of obesity ("groupobese") in the coefficient table of the linear model, except it has the opposite sign. This sign is arbitrary and simply reflects which mean is subtracted from which. The p-value for both is the same.

9.2.1.2 A student t-test is *not* equivalant to the t-value and p-value in a coefficient table of a linear model *if* there are more than two levels in the treatment factor. This is a feture of a linear model, not a bug.

Let's return to the vole data. The t and p values of the effects of vitamin C and vitamin E in the coefficient table of the linear model of $lifespan \sim treatment$ are

Level

t

p

vitamin_C

-2.113

0.037

vitamin_E

-1.713

0.090

while the t-tests between the two supplement levels and the control are

Level

t

p

vitamin_C

-1.981

0.051

vitamin_E

-1.628

0.108

The *t*-test statistics differ from those of the linear model because the two use *different* standard errors in the denominator of *t*. Both denominators are computed from a **pooled variance**, which estimates the population variance using a weighted average of the variances of each of the groups in the model. The linear model contains all three levels (groups) of *treatment* and, consequently, the pooled variance is computed from the variances of all three groups. The *t*-test uses the pooled variance averaged over only the two levels compared.

If the linear model uses a pooled variance over all three levels, this raises the question of why the standard error of the vitamin C and vitamin E effects differs (see the full table above). The reason is the vitamin C and vitamin E groups have different sample sizes, so while the standard errors in the table are computed using a common variance, they are computed using different n.

9.2.1.3 Feature not a bug

9.2.1.4 Use the linear model, not a t-test.

9.2.2 ANOVA is a special case of a linear model

9.3 Working in R

Import the vole data from the Dryad repository using the information above and in Chapter 2 section Vole data.

9.3.1 Fitting the model

As with a single, continuous X, we fit the model using the `lm` function and with the model formula of the form `y ~ x`. Note that the R formula can use the single categorical variable *treatment*. The code underneath lm will note that *treatment* is a factor with three levels and will automatically create the two dummy variables noted above in the linear model.

```r
fit <- lm(lifespan ~ treatment, data=vole)
```

All of the same scripts to access the information in `fit` that we used with the continuous X analysis are the same. For example, the base R `summary` function gives the same information as in the continuous X example. Other useful functions on the lm object ("fit") are

1. `coefficients(fit)` and `coefficients(summary(fit))`. Note the difference between these. The first is useful if we just want to extract the coefficient. The second if we want the addtional information. These can both be shortened using `coef` in place of `coefficients`.

Let's look at the coefficient table

```r
coef(summary(fit))
```

```
##                      Estimate Std. Error    t value      Pr(>|t|)
## (Intercept)         503.39286   27.40978 18.365445 1.078296e-32
## treatmentvitamin_C -115.07707   54.45772 -2.113145 3.726632e-02
## treatmentvitamin_E  -89.91667   52.48574 -1.713164 9.001428e-02
```

The reference level is "control" – we know this because there are estimates of the effects for the other two levels.

9.3.2 Changing the reference level

R assigns the order of the levels of a factor alphabetically, so the order of the levels of treatment are "control", "vitamin_C", "vitamin_E". The first of these is the reference level. Remember the intercept is the mean of the reference group and the remaining estimates are the differences in means from this reference. If we want to make some other level the reference, we can change the order of the factor levels using

```r
vole[, treatment:=factor(treatment,
                    levels=c("vitamin_C", "vitamin_E", "control"))]
```

The order of the levels in the levels argument sets the new order for any further analysis. Refit the model to see how this re-ordering changes the coefficients

```r
fit2 <- lm(lifespan ~ treatment, data=vole)
coef(summary(fit2))
```

```
##                     Estimate Std. Error   t value      Pr(>|t|)
## (Intercept)         388.3158   47.05685 8.2520573 1.005619e-12
## treatmentvitamin_E  25.1604   64.94462 0.3874132 6.993356e-01
## treatmentcontrol   115.0771   54.45772 2.1131453 3.726632e-02
```

Understand why the values of these coefficients differ from those in the coefficient table above.

Here, I'm returning the factors back to the original order.

```
# put factors back to original
vole[, treatment:=factor(treatment,
                         levels=c("control", "vitamin_C", "vitamin_E"))]
```

9.3.3 An introduction to contrasts

We often want to compare more than just the non-reference levels to the reference level. For example, we might want to compare the effects of the vitamin E supplementation to vitamin C supplementation. Or, we might want to combine (or "pool") vitamin C and vitamin E levels effects into a single "anti-oxidant" level and compare to the control. These comparisons of means are called linear **contrasts**. The emmeans package is a good package for obtaining contrasts for both simple linear models computed with `lm` and for more complicated statistical models. If you haven't already, download the emmeans package.

```
fit.em <- emmeans(fit, specs="treatment")
fit.em
```

```
##  treatment emmean   SE df lower.CL upper.CL
##  control      503 27.4 93      449      558
##  vitamin_C    388 47.1 93      295      482
##  vitamin_E    413 44.8 93      325      502
##
## Confidence level used: 0.95
```

The `emmeans()` function returns various estimated means, depending on what is specified with the `spec=` parameter. Here the grouping variable "treatment" is specified, so the means returned are estimates of $E(lifespan|treatment)$, the modeled means for each level of treatment. For this simple analysis, the modeled means are simply the group means. Note that the default value returned is a table with the standard error and 95% confidence limits of the estimates.

Let's use the emmeans object to get the contrasts for all combinations of treatment levels.

```
summary(contrast(fit.em, method="revpairwise", adjust="none"), infer=c(TRUE, TRUE))
```

```
##  contrast             estimate   SE df lower.CL upper.CL t.ratio p.value
##  vitamin_C - control    -115.1 54.5 93     -223    -6.93  -2.113  0.0373
##  vitamin_E - control     -89.9 52.5 93     -194    14.31  -1.713  0.0900
##  vitamin_E - vitamin_C    25.2 64.9 93     -104   154.13   0.387  0.6993
##
## Confidence level used: 0.95
```

1. method="revpairwise". `contrast` can create different combinations of differences between means. Here I've specified all pairwise differences (the

"rev" reverses the order of the subtraction). Notice that the statistics (estimate, SE, etc) are equal to the same statistics for b_1 and b_2 of the linear model. I said earlier that these coefficients are contrasts!

2. adjust="none". In classical frequentist hypothesis testing, the p-value of a contrast in what are called "post-hoc tests" is adjusted to reflect "multiple testing" (more than one p-value is being computed). This adjustment is almost standard in biology, but the practice is hugely controversial. The concept of multiple testing is important, and we will return to this in a future chapter, but here I have chosen to show the unadjusted p-value. The reason is that I want the unadjusted confidence interval and the adjustment would adjust these as well. If deleted `adjust="none"` from the script, the contrast function would default to the **Tukey HSD** (Honestly Significant Difference) test. There are literally dozens and dozens of post-hoc tests, which largely reflects the misplaced emphasis on "better" *p*-values rather than parameter estimates and their uncertainty.

3. infer=c(TRUE, TRUE). This parameter controls what kind of inference to put in the table. The first value specifies the inclusion of the CI (emmeans uses "CL" for confidence limit), the second value specifies the inclusion of *t* and *p*-values.

9.3.4 Harrell plot

9.3.4.1 Installing the harrellplot package

The harrellplot package is available on github but not a cran repository and, therefore, takes a little more work to install. To install a package from a github repository, 1. load library(devtools) – this may need to be installed first using the R Studio Tools > Install Packages... tool 2. install harrellplot from github. In the console, type

```
install_github("middleprofessor/harrellplot")
```

3. load the harrellplot package
4. harrellplot requires other packages including broom, Hmisc, car, lme4, and lmerTest. If you haven't installed these do. load these with the library() function at the start of your notebook.

9.3.4.2 Using harrellplot to make a nice, publishable plot of treatment effects

In the console type `?harrellplot` to see the many parameters. Unlike ggplot2, variable names need to be specified with quotes in the harrellplot function. The harrellplot function is a list with several elements.

Here is the default plot

```
vole.harrellplot <- harrellplot(x="treatment", y="lifespan", data=vole)
vole.harrellplot$gg # gg is the plot object
```
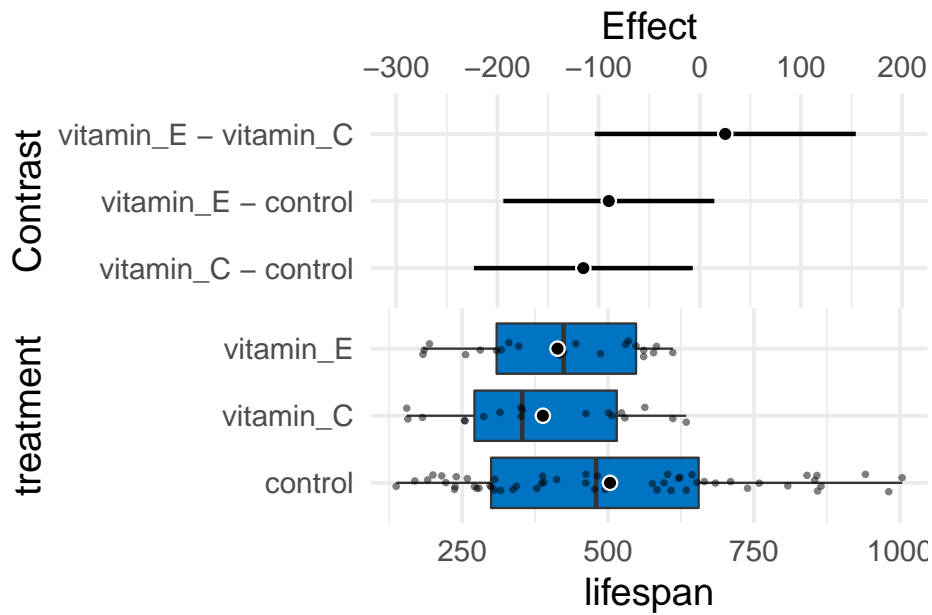
Chapter 10

Model Checking

```r
# a function to transform a vector into quantiles
# not if the data are 1:n then the output is "rankits"
quantilize_1 <- function(x){
  # this is the ppoints(x) function
  m <- length(x)
  s <- trunc(rank(x))
  a <- ifelse(m <= 10, 3/8, 1/2)
  q <- (s-a)/(m + (1-a) - a)
  return(q)
}
```

10.1 Do coefficients make numeric sense?

10.2 All statistical analyses should be followed by model checking

We us a linear model (or statistical model more generally) to infer effects or predict future outcomes. Our inference is uncertain. Given some modeling 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 Shaprio-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 increses, 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.

10.3   Linear model assumptions

Assumptions of a linear model concern the distribution of the "random draw" in the underlying statistical model. Again, in the random error specification of a linear model

$$Y = \beta_0 + \beta_1 X + \varepsilon \qquad (10.1)$$

$$\varepsilon \sim N(0, \sigma) \qquad (10.2)$$

the random draw (the "error") is from a normal distribution with mean zero and standard deviation σ. In the random conditional response specification

$$y_i \sim N(\mu_i, \sigma) \tag{10.3}$$
$$\mathrm{E}(Y|X) = \mu \tag{10.4}$$
$$\mu_i = \beta_0 + \beta_1 x_i \tag{10.5}$$

the random draw is a value drawn from a normal distribution with mean $\mu_i = \beta_0 + \beta_1 x_i$ and variance σ^2. Any inference about the parameter β_1 (such as confidence intervals or hypothesis tests) assumes that the these distributions are IID Normal where IID is **independent and identically distributed** and Normal refers to the Normal (or Gaussian) distribution.

1. Independent means that the error for one case cannot be predicted from the error of any other case. This 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 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 (10.2).

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**. 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 (10.2).

2. Identical means that the errors are "drawn" from the same distribution. Since the model is a linear model, this distribution is a Normal distribution with mean zero and variance σ^2. A consequence of "identical" is that the error variance is **homoskedastic**, or constant, or independent of X. If the error variance differs among the X then the errors are **heteroskedastic**. 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. 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), which is an extension of a linear model. Heteroskedasitc error arising for other reasons, both biological and experimental, can be modeled with Generalized Least Squares (GLS) or with linear mixed models..

3. Normal (Gaussian) error means that 1) the response is continuous and 2) the probability of sampling an individual measuring 0.5 units below the population mean is the same as the probability of sampling an individual measuring 0.5 units above the population mean. Counts (number of cells, number of eggs, number of mRNA transcripts) and binary responses (sucessful escape or sucessful infestation of host) are not continous and often often have asymmetric probablity 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 a **generalized linear model** (GLM), which, again, is an extension of the linear model in equation (10.2).

10.4   Diagnostic plots use the residuals from the model fit

10.4.1   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 (compaactly 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 (10.2). This *is not* true for generalized linear models because GLMs are not specified using (10.2).

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 (10.2) 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 general way of thinking about the assumed distribution uses the specification in model (10.5), 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 response 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

model-check-residuals1-1.bb

A

B
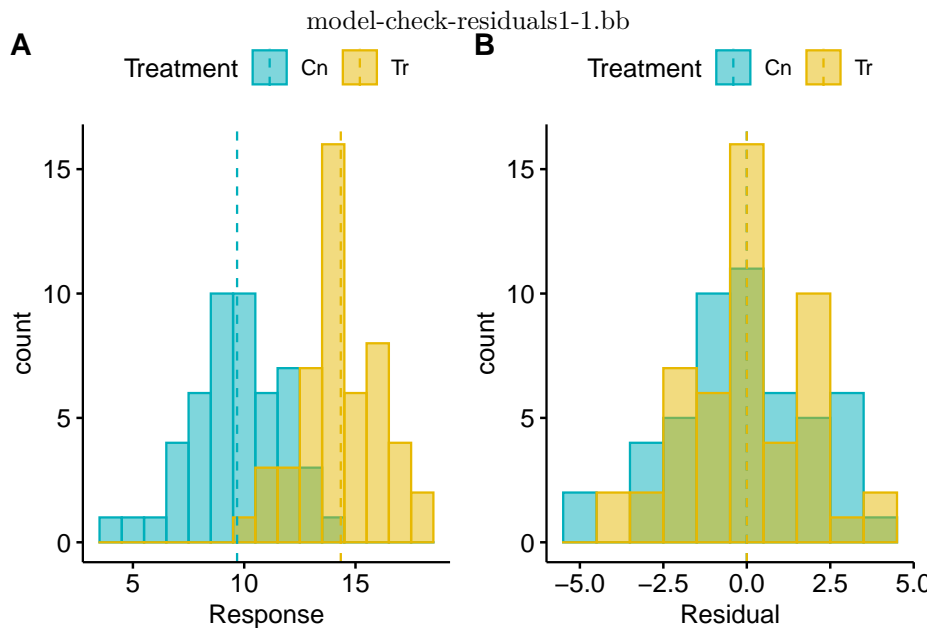
Figure 10.1: (#fig:model-check-histogram, model-check-residuals1)Histogram of the (A) response, showing with modes near the true means of each group and (B) residuals, with a mode for both groups at zero.

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 distincly 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".

10.4.2   A Normal Q-Q plot is used to check normality

A Normal Q-Q plot of the residuals can be used to check how closely the residuals approximate a normal distribution. 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 (doing this makes not 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.

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 distribtion – 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.

If the sampled distribution closely approximates 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 a normal distribution. Because of sampling, the sampled values always deviate from the line, especially at the ends. If the sample was drawn from a normal distribution, these deviations should be small if the sample size is big, but can be more pronounced with a small sample size. This makes it hard to have much confidence in the "normality" of a small sample.

Let's have a look at a Normal Q-Q plot of the residuals of the fake data generated above.
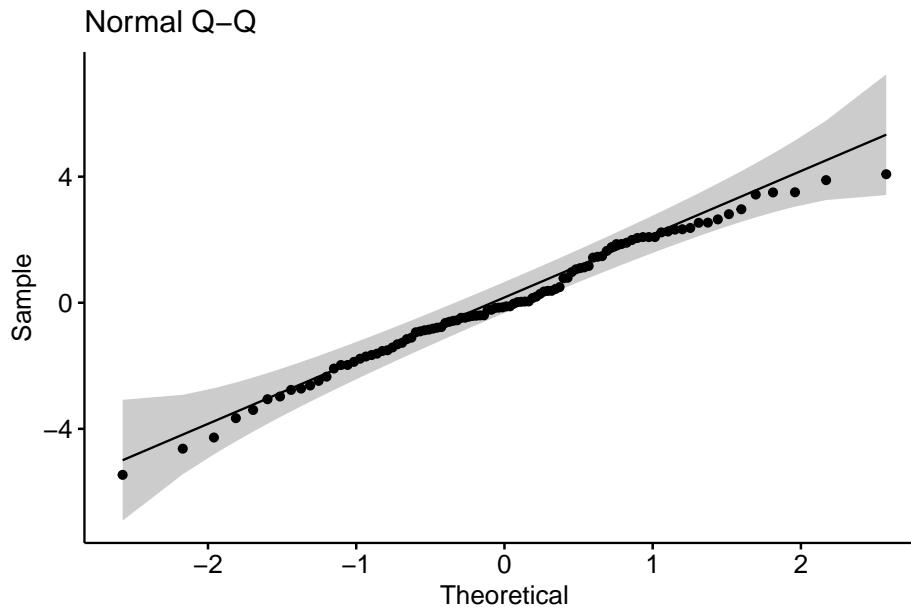
Rules of a Q-Q plot

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. Should we fit a different model given these deviations? To answer this, we look look at the shaded area, which represents the range of expected deviations from expectation (the line) given the sample size. Clearly the deviations are within this range.

Now let's look at simulated samples drawn from non-normal distributions to identify their characteristic deviations.

10.4.2.1   Right skewed

Many biological measures are from a distribution with long, right tails (right skewed). Examples include many count variables (number of eggs in a clutch, number of cells colonized by a parasite), and measures of time, weight, or length. What is common to all of these is unconstrained upper bounday but a constrained lower boundary at or above zero (A nest might have zero but eggs. The weight of a fat depot must be greater than zero but the weight of a specific species of fish in a trawl catch might be zero).

A long right tail of conditional responses creates a characteristic positive deviation of the largest quantiles in a Normal Q-Q plot of the residuals from a linear model. Positive deviations at the upper end indicate larger values than

expected given a normal distribution. This is the signature of the residuals of a
linear model fit to right skewed data.
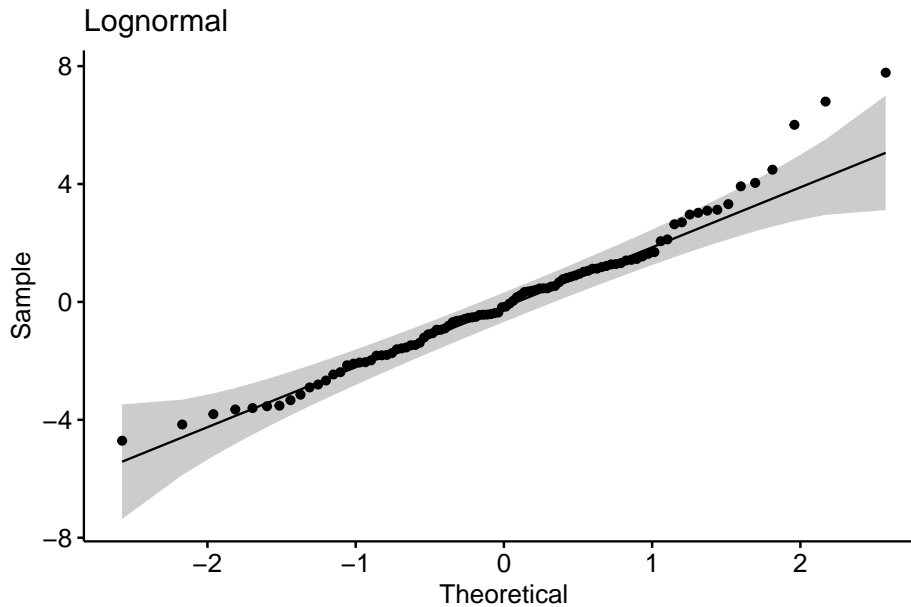
A continous response with a right skewed distribution can be modeled with a
generalized linear model using a lognormal or gamma distribution. A count
response can be modeled with a generalized linear model using a Poisson, quasi-
Poisson, or negative binomial distribution (Chapter xxx).

10.4.2.2   Excess zeroes

Count data often have an excess of zeroes (for example a lake with no nests or
a host with no parasites), resulting in positive deviations (closer to zero) at the
lower end of the quantile range.

```r
# zero inflated
mu_pois <- 2.7
beta_pois <- 2
p_zero <- 0.3

fd.zipois <- data.table(
  Treatment=rep(c("Cn", "Tr"), each=n),
  Response=c(SpiecEasi::rzipois(n, lambda = mu_pois, pstr0 = p_zero), SpiecEasi::rzipoi
)
m1 <- lm(Response ~ Treatment, data=fd.zipois)
fd.zipois[, Residual:=residuals(m1)]
gg1.zipois <- gghistogram(data=fd.zipois, x = "Response",
```

```
                                color="Treatment", fill="Treatment",
                                add = "mean", rug = FALSE,
                                bins=9,
                                palette = c("#00AFBB", "#E7B800")
)

gg3.zipois <- ggqqplot(data=fd.zipois, x = "Residual", title="zero-inflated poisson")

gg3.zipois
```
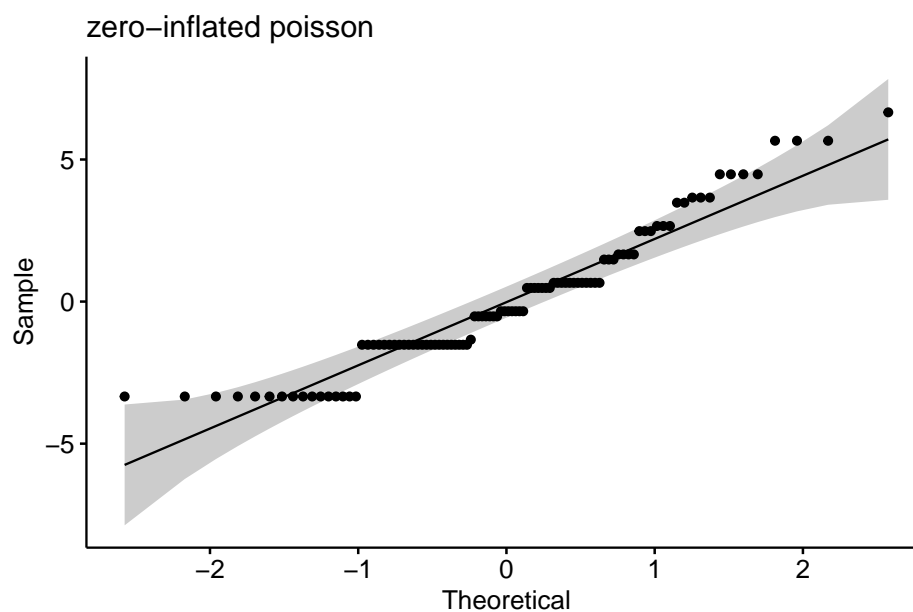

10.4.2.3 Constrained lower and upper bounds

Proportions are constrained to values between zero and one. A proportion can have a distribution that approximates a normal distribution if the mean is near 0.5 and the standard deviation is small. But, more generally, proportions can have distributions with diverse shapes.
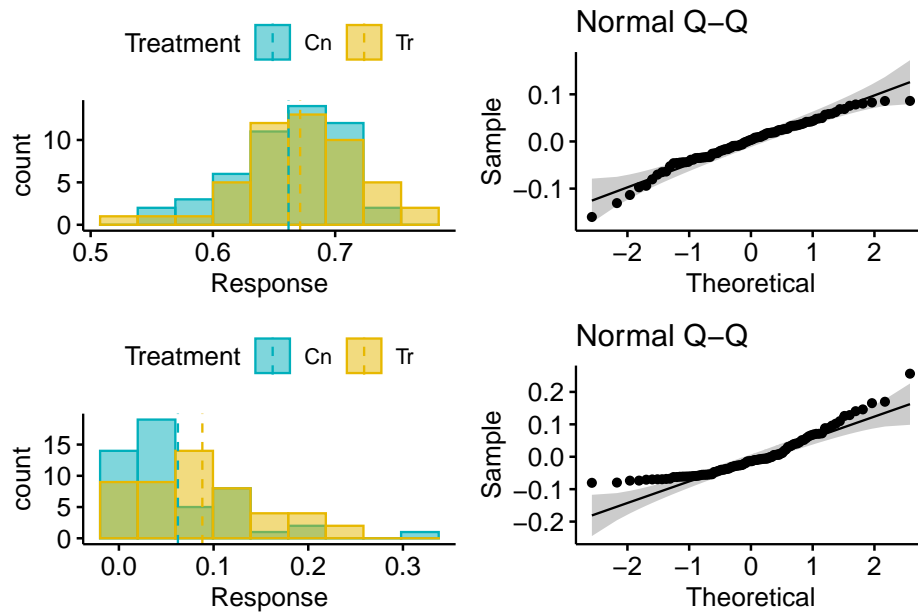
10.4.2.4   Binary responses

10.4.3   Outliers - an outlier is a point that is highly unexpected given the modeled distribution.

10.5   Model checking homoskedasticity

10.6   Model checking independence - hapiness adverse example.

10.7   Using R

Chapter 11

Model Fitting and Model Fit (OLS)

11.1 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.

11.2 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 elipse 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 it's 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

$$b_1 = \frac{\text{COV}(X, Y)}{\text{VAR}(X)} \tag{11.1}$$

This equation is worth memorizing. We will generalize this into a more flexible equation in a few chapters.

11.3 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 highyly 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*).
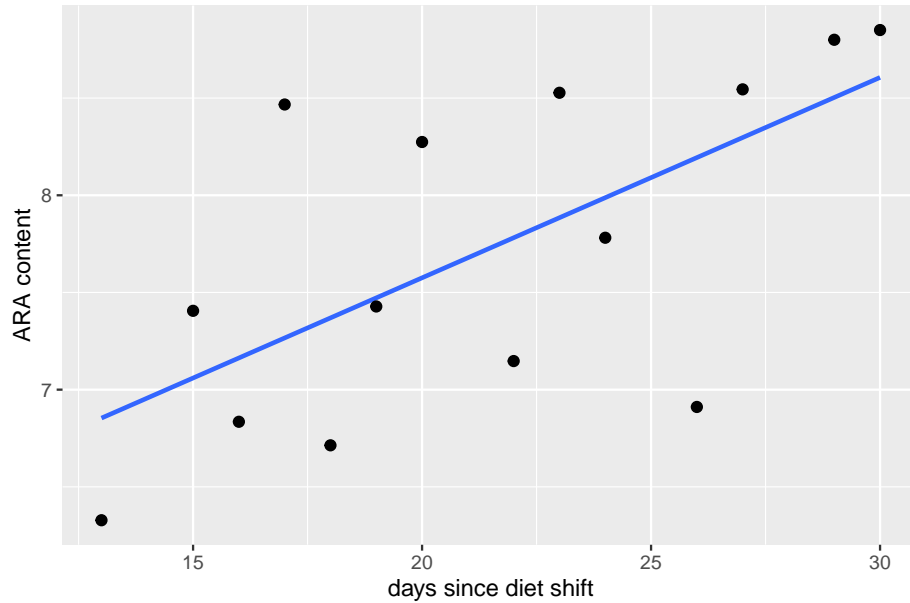
The statistic R^2 is a measure of the fit of a model to data. The R^2 for the fit of the egg data is 0.42. R^2 is the fraction of two variances $\frac{\text{VAR}(Model)}{\text{VAR}(Y)}$, or, the fraction of the variance of Y "explained by the model." The value of R^2 ranges from zero (the fit cannot be any worse) to one (the fit is "pefect").

To understand R^2, and its computation, a bit more, let's look at three kinds of deviations.

Figure 11.1A 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 $ARA_E GG_M G$. Figure 11.1B 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 11.1C shows the deviations of the modeled values to the mean value. The bigger these deviations are, the better the model fit.

The sums of the squares of these deviations (or "sums of squares") have names:
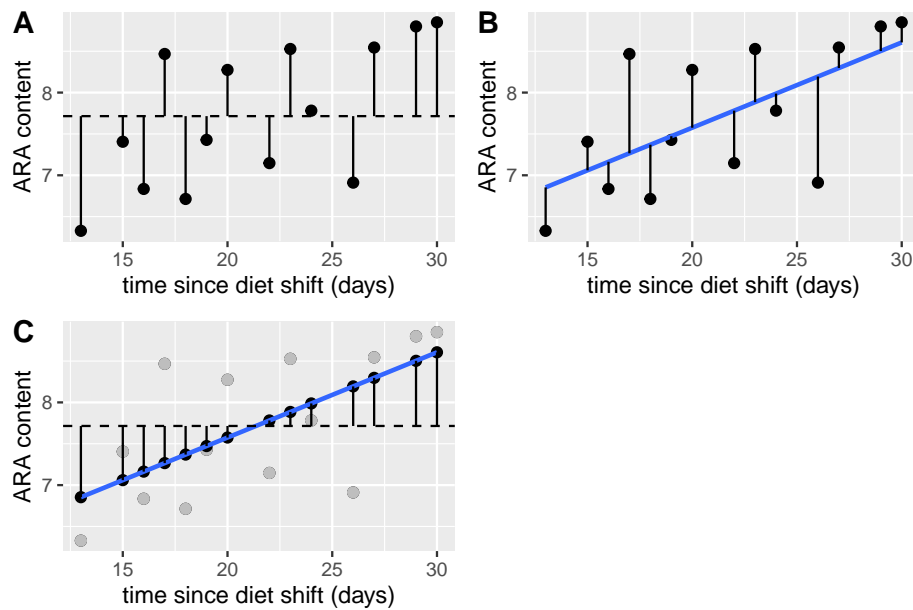
Figure 11.1: Three kinds of deviations from a fit model. A. Deviations of the measured values from the mean. These are in the numerator of the equation of the sample variance. The dashed line is the mean ARA content. B. Deviations of the measured values from the modeled values. The sum of these deviations 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

$$\text{SS(total)} = \sum (y_i - \bar{y})^2 \tag{11.2}$$

$$\text{SS(error)} = \sum (y_i - \hat{y}_i)^2 \tag{11.3}$$

$$\text{SS(model)} = \sum (\hat{y}_i - \bar{y})^2 \tag{11.4}$$

Again, SS(total) is the numerator of the equation for the sample variance. It is called "s-s-total" because SS(total) = SS(model) + 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

$$R^2 = \frac{SS(model)}{SS(total)} \tag{11.5}$$

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

$$R^2 = 1 - \frac{SS(error)}{SS(total)} \tag{11.6}$$

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 12

Best Practices – Issues in Inference

12.1   t-tests and ANOVA

Welch, paired

12.2   Power

12.2.1   "Types" of Error

I, II, S, M

12.3   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.

12.3.1　Some background

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 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 probabilty 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

12.3.2 Multiple testing – working in R

12.3.2.1 Multiple pairwise contrasts

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

12.3.2.2 Tukey HSD adjustment of all pairwise comparisons

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
```

12.3.2.3 Tukey HSD adjustment of conditional effects

Here I compare the Tukey HSD adjustment between all pairwise comparisons and the subset that are the conditional (simple) effects.

12.4 p-hacking

12.5 difference in p is not different

12.6 Inference when data are not Normal

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 "the data" (specifically, the conditional values, 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 cancan choose among several strategies

1. continue to use the linear model, many of which are fairly robust to non-normal data, especially when the sample size is not small.
2. transform the data in a way that makes the conditional response more closely approximate a normal distribution.
3. use a non-parametric method, which are methods that do not assume a particular distribution
4. use a generalized linear model (GLM), which is appropriate if the conditional response approximates any of the distributions that can be modeled using GLM.

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).

12.6.1 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 co-efficient or contrast but this itself produces problems. For example, the backtransformed mean of the log-transformed response is not the mean on the origianl 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?

12.6.2 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-Wilcoxan
   - Paired t test: Wilcoxan 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 supplment to, and not replacement of, inference from a generalized linear model.

12.6.2.1 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.

Create data from a negative binomial distribution, which is right skewed

```r
set.seed(5)
n <- 10
fake_data <- data.table(
  treatment = rep(c("cn", "tr"), each=n),
  count = rnegbin(n*2, mu=rep(c(10, 14), each=n), theta=1)
)
```

Some statistics of the fake data

```r
fake_data[, .(N=.N, mean=mean(count), sd=sd(count)), by=treatment]
```

```
##    treatment  N mean        sd
## 1:        cn 10  7.1  8.372574
## 2:        tr 10 18.5 16.029487
```

A plot of the fake data

```r
ggerrorplot(x="treatment",
            y = "count",
            data = fake_data,
            add = c("mean_ci", "jitter"),
            palette = "jco",
            color = "treatment")
```
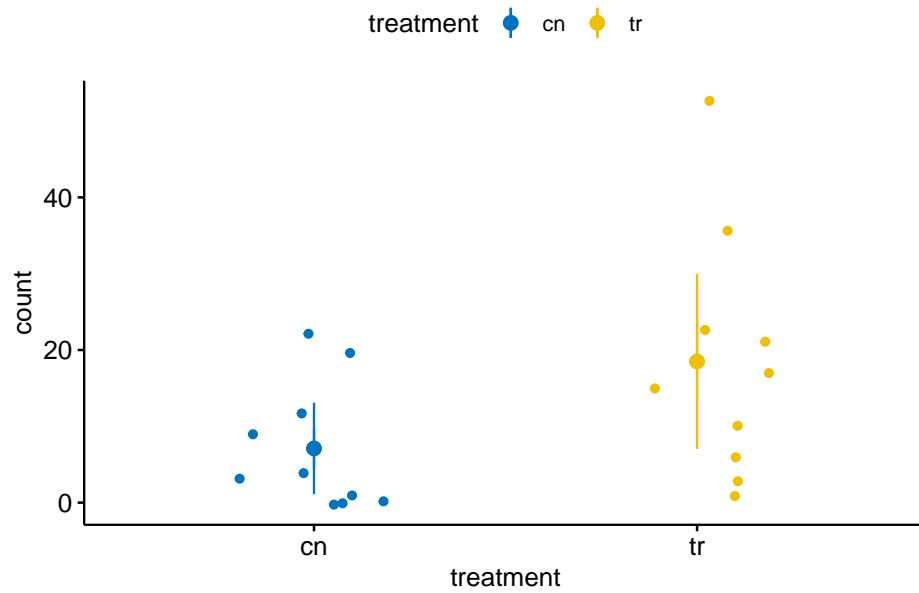
What the distribution of the fake data looks like

```
n <- 10^4
qplot(rnegbin(n, mu=10, theta=1))
```
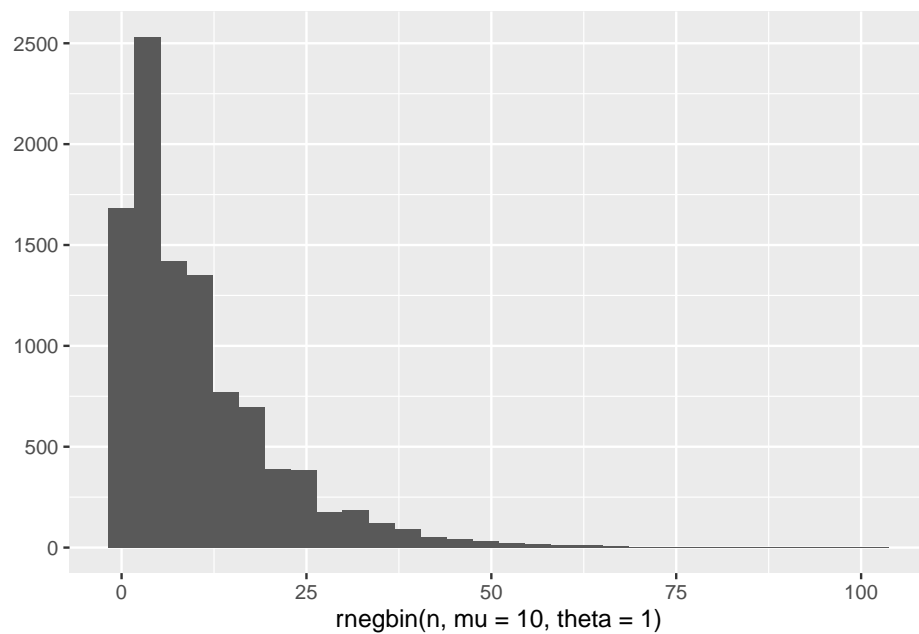

A linear model to estimate the treatment effect.

```
knitr::kable(coef(summary(lm(count ~ treatment, data=fake_data))),
             digits = c(1, 1, 1, 3))
```

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

7.1

4.0

1.8

0.096

treatmenttr

11.4

5.7

2.0

0.062

A non-parametric (Mann-Whitney-Wilcoxon) test

```
wilcox.test(count ~ treatment, data=fake_data)
```

```
## Warning in wilcox.test.default(x = c(0L, 1L, 4L, 20L, 0L, 9L, 3L, 12L,
## 22L, : cannot compute exact p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  count by treatment
## W = 24, p-value = 0.05336
## alternative hypothesis: true location shift is not equal to 0
```

Finally, a randomization, or permutation, test. The test statistic is the difference in means. The test is two-tailed, so the absolute difference is recorded. The first value computed is the observed absolute difference. The *p*-value is computed as the relative frequency of values greater than or equal to this observed value.

```
set.seed(1)
n_permutations <- 5000
d <- numeric(n_permutations)
fake_data[, count_perm := count]
```

```r
for(i in 1:n_permutations){
  d[i] <- abs(mean(fake_data[treatment=="tr", count_perm]) -
    mean(fake_data[treatment=="cn", count_perm]))
  # permute the count_perm column for the next iteration
  fake_data[, count_perm := sample(count)]
}
p <- sum(d >= d[1])/n_permutations
p
```

```
## [1] 0.0642
```

12.6.2.2   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 due to the term divided by the residual sum of squares of the model.

```r
coef(summary(lmp(count ~ treatment, perm="Prob", Ca=0.01,
                 data=fake_data)))
```

```
## [1] "Settings:   unique SS "

##               Estimate Iter Pr(Prob)
## (Intercept)       12.8 5000   0.0642
## treatment1        -5.7 5000   0.0642
```

12.6.3   Performance of parametric, classic non-parametric, and permutation test

12.6.3.1   Type I error

```r
# 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", "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, "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      mww     perm
## 0.04300 0.04525 0.04825
```

All three 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.

12.6.3.2 Power

```
# 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=20, theta=1), nrow=n))
treatment <- rep(c("cn", "tr"), each=n)

tests <- c("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, "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      mww     perm
## 0.22600 0.21225 0.25600
```

At this sample size, and for the modeled fake data, the permutation test has
about 13-15% more power than the linear model and Mann-Whitney-Wilcoxan
test.

12.7   max vs. mean

12.8   pre-post, normalization

Chapter 13

Plotting Models

So, along the lines of Sarah Susanka's "Not So Big House," Kolbert asks the group, "What would a Pretty Good House look like?" – Michael Maines[1]

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 all 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.

13.1   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 non-bacterial factors such as host pro-inflammatory cytokines rather than microbial derived factors cause transmissible intestinal inflammation"), and sox10.

13.1.1   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.

---

[2]Rolig, 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
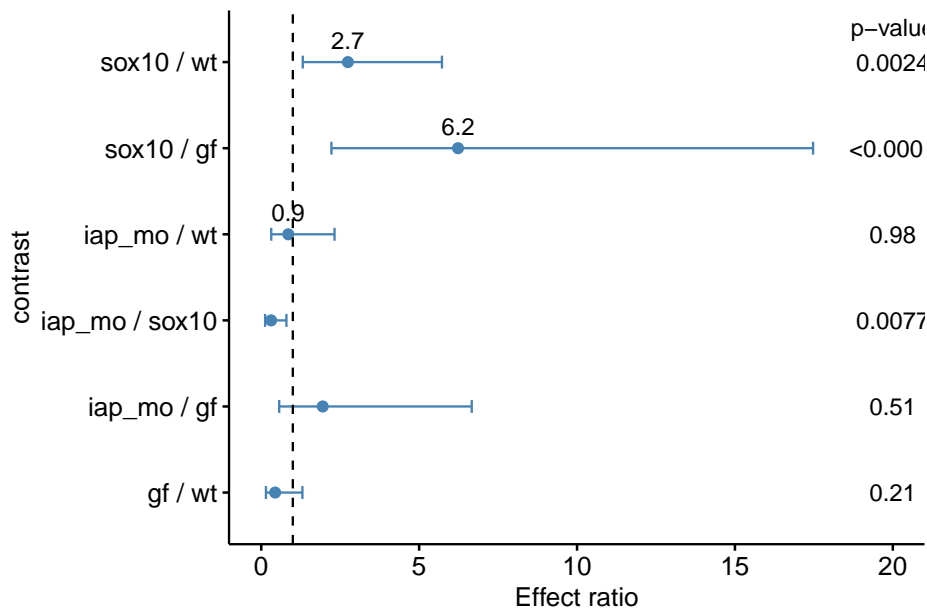
Figure 13.1: Effects Plot

The y-axis contains each of the *paired comparisons* among the four treatment levels. The x-axis is the response, which is the ratio of the means of the two groups 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 disscussed 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.

13.1.2 Pretty good plot component 2: Modeled mean and CI plot with jittered raw data

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 unusal values. The data used in Fig 13.1 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.
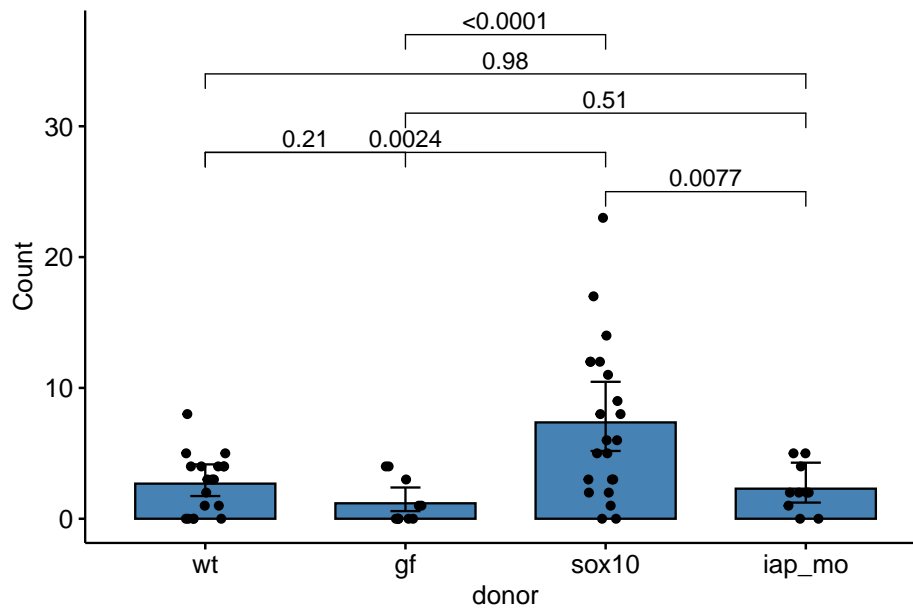
Figure 13.2: Mean and error plot

Fig. 13.2 plots the *modeled* means, represented by the tops of the bars, the modeled 95% confidence intervals of each mean, represented by the error bars, 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, or the researchers are using a *hierarchical 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 means that are relevant to inference.

Fig. 13.2 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. 13.2 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.

13.1.3 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.

13.2 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/Violoin/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
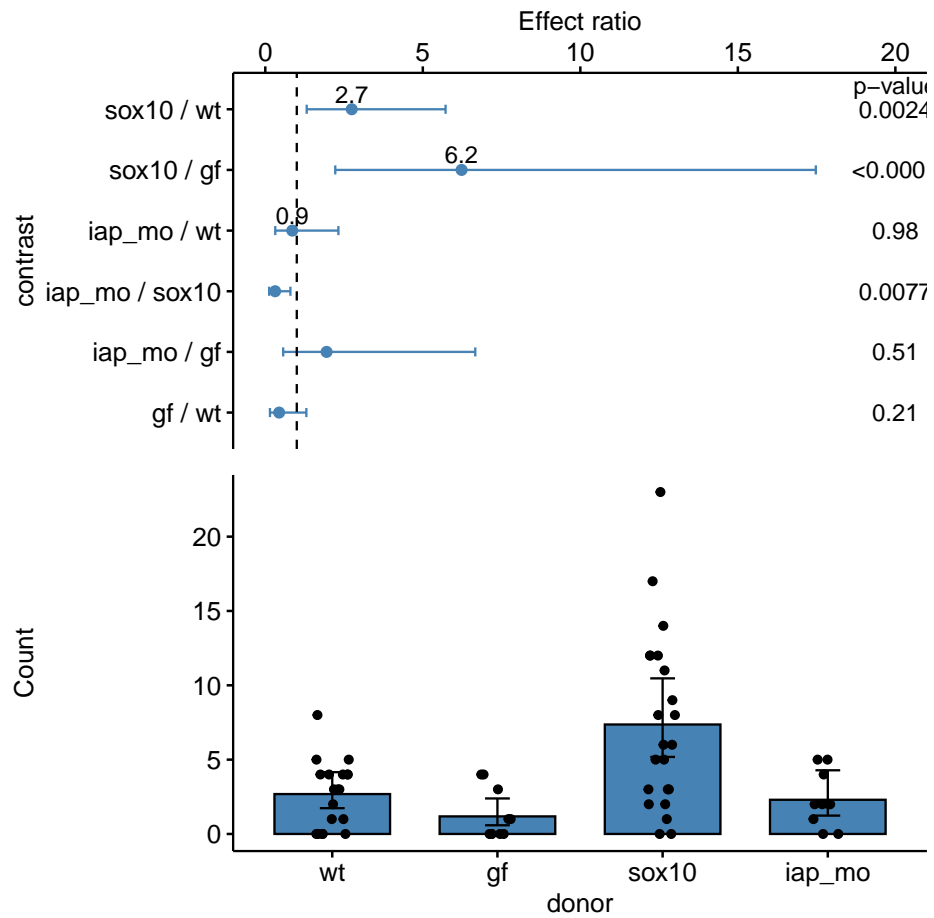
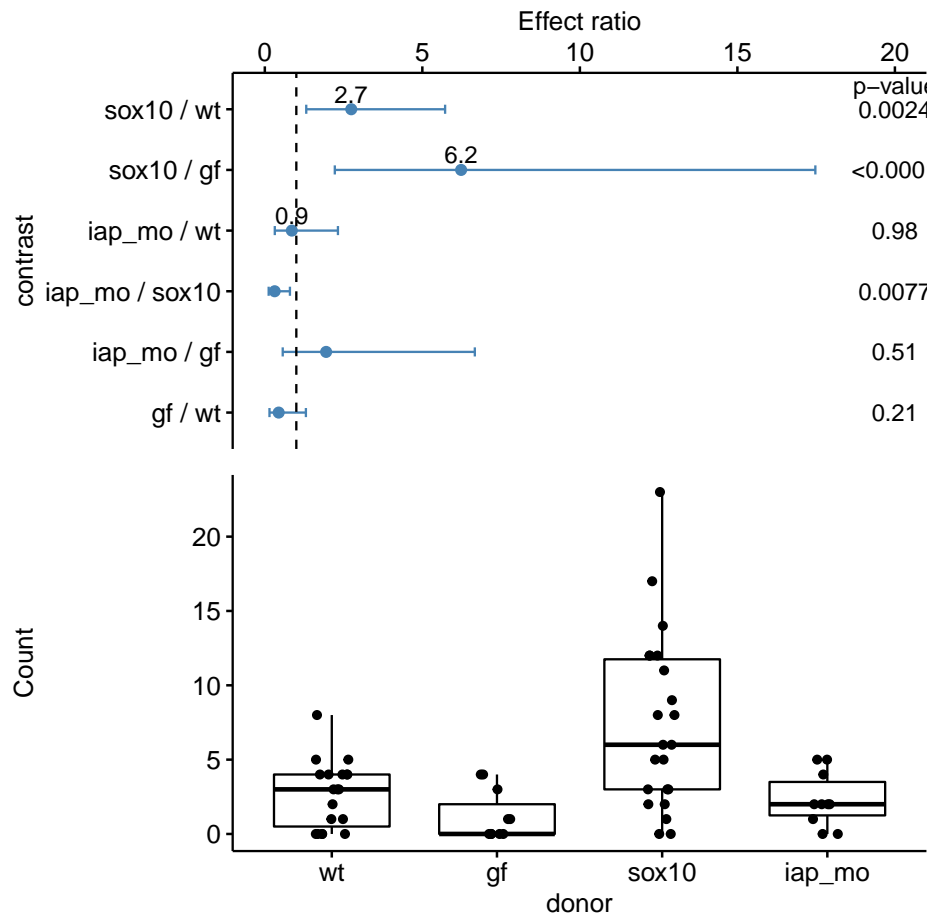Figure 13.3: A pretty good plot

Figure 13.4: Another pretty good plot

a response that cannot be negative, such as a count. Even if the standard 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.

13.3   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) 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. That said, the means and error intervals used in `ggpubr` plots are the raw and not modeled values, and, consequently, `ggpubr` is not sufficient to generate pretty good plots. It is easy enough to add custom error bars to a ggpubr plot.

13.3.1   Unpooled SE bars and confidence intervals

`ggplot2` and `ggpubr` default to unpooled error intervals (standard error bars and confidence intervals).
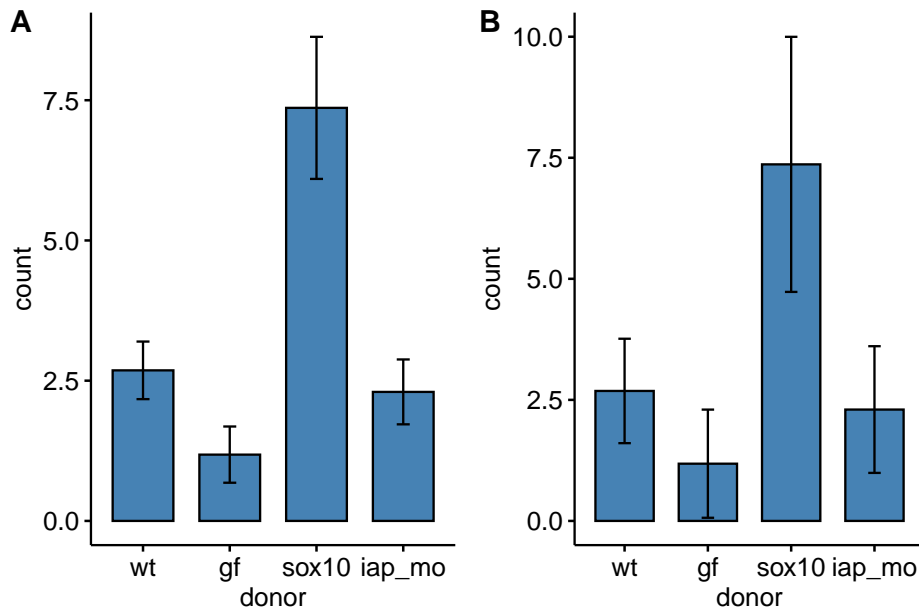
Figure 13.5: (A) Mean and 1 SE error bar. (B) Mean and 95% CI.

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

13.3.2 Adding bootstrap intervals

```
gg <- ggbarplot(x="donor",
                y="count",
                data=exp2d,
```

```
              add=c("mean"),
              color = "black",
              fill = "steelblue"
) +
  stat_summary(fun.data = "mean_cl_boot", geom = "errorbar", width=0.1)
gg
```
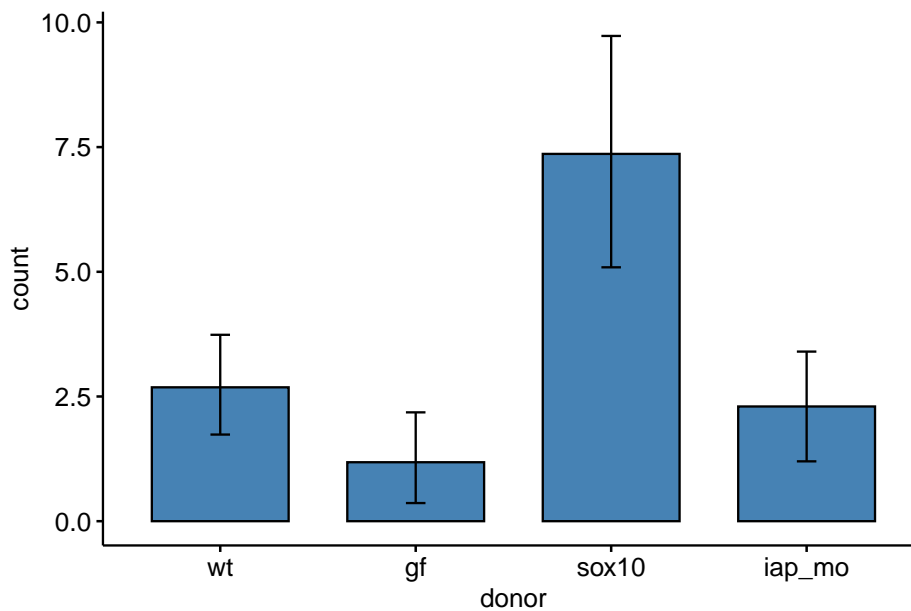

13.3.3   Adding modeled error intervals

Plotting the modeled error intervals "shows the model". `emmeans` is a comprehensive and flexible package for computing modeled standard errors and confidence intervals for all of the statistical models covered in this text.

```
m1 <- glm.nb(count ~ donor, data=exp2d)
emm.m1 <- emmeans(m1, specs="donor", type="response")
effects.m1 <- summary(contrast(emm.m1, method="revpairwise"), infer=c(TRUE, TRUE))

# make emm.m1 a data.table
emm.m1 <- data.table(summary(emm.m1))
# the y column needs to have the same label as the plotted data
setnames(emm.m1, old="response", new="count")
```

13.3.3.1 Modeled error intervals of the effect

```r
(gg1 <- ggdotplot(x="contrast",
                  y="ratio",
                  data=effects.m1,
                  color = "steelblue",
                  fill = "steelblue",
                  size=0.5) +

  # add either the SE or CI, contained in effects.m1
  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)
```
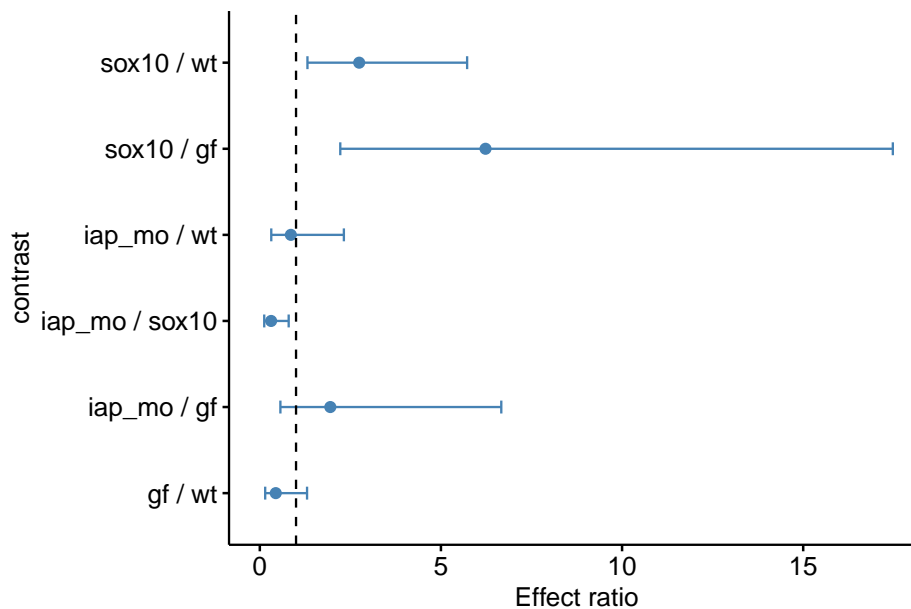

13.3.3.2 Modeled error intervals of the mean

```r
(gg2 <- ggbarplot(x="donor",
         y="count",
         data=exp2d,
         add=c("mean", "jitter"),
```

```
            color = "darkgray",
            fill = "steelblue",
            size=0.5) +
  ylab("Neutrophil count") +

  # emm.m1 contains the SE and 95% CIs. Either could be plotted. Here I plot the CI
  geom_errorbar(data=emm.m1, aes(ymin=asymp.LCL, ymax=asymp.UCL), width=0.1) +
  NULL)
```
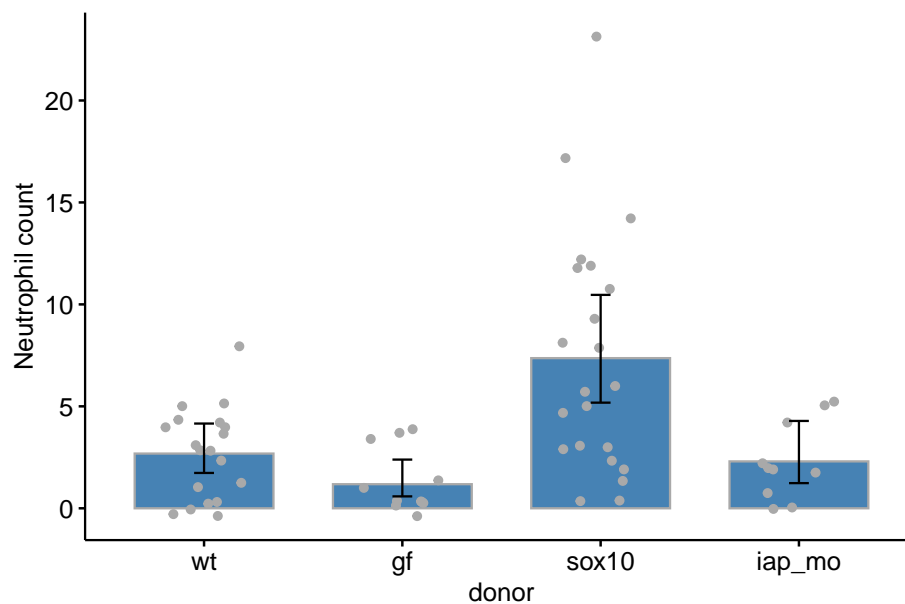

Note that the CIs are asymmetric about the mean (and these modeled CIs will never include negative values for count data).

13.3.3.3 Combining effects and response plots

The ggplots are combined using the package `cowplot`

```
gg_top <- gg1 + scale_y_continuous(position="right")
plot_grid(gg_top, gg2, nrow=2, align = "v", rel_heights = c(1, 1))
```
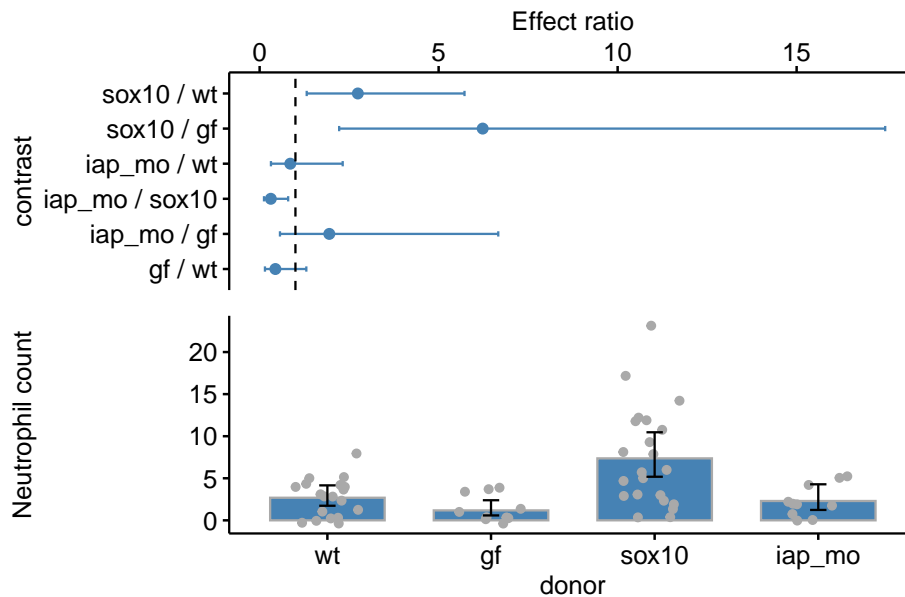
13.3.4 Adding p-values

p-values are added to the base ggpubr plot using `stat_compare_means`. The pairs to compare are specified with `comparison =`. The model to compute the p-values is "t.test". **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 (which adjusts the standard error to account for heterogenity in variance between groups). In this example, there are multiple tests and the standard error could be the pooled estimate estimated from the linear model, or a pairwise estimate. 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.

```r
pairs_i <- list(c("sox10", "iap_mo"), c("sox10", "gf"), c("sox10", "wt"))
ggbarplot(x="donor",
          y="count",
          data=exp2d,
          add=c("mean_se", "jitter"),
          color = "black",
          fill = "steelblue",
          size=0.5) +
  stat_compare_means(method = "t.test", comparisons=pairs_i) +
  ylab("Neutrophil count") +
```

NULL
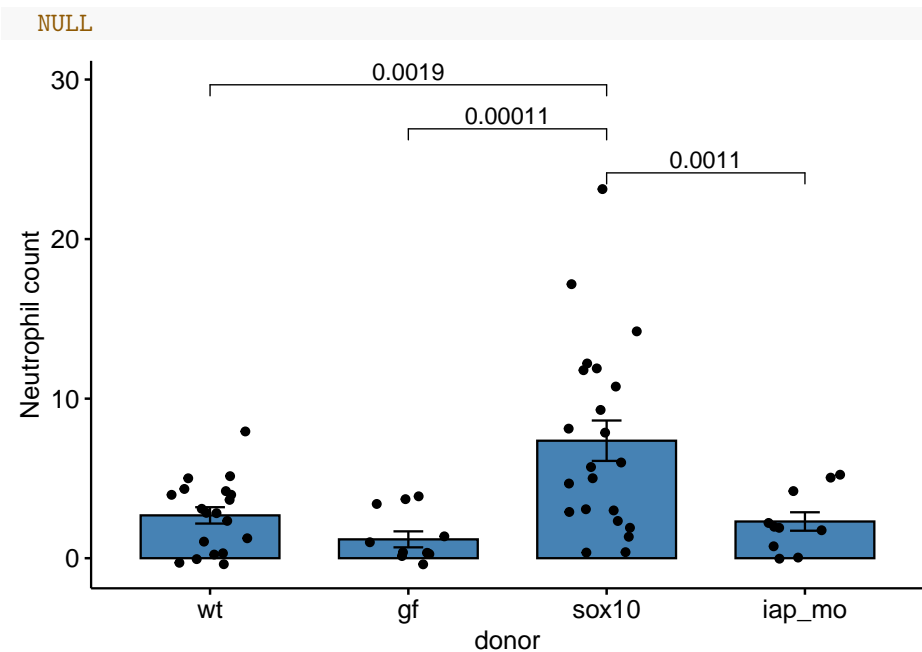

```
# checks on the p-value
# t-tests using SE pooled over all four groups
check_it <- FALSE
if(check_it==TRUE){
  m1 <- lm(count~donor, data=exp2d)
  m1.emm <- emmeans(m1, specs="donor")
  contrast(m1.emm, method="trt.vs.ctrl", ref=3, adjust="none") # not pooled SE

  pairwise.t.test(exp2d$count, exp2d$donor, p.adjust.method="none", pool.sd=FALSE) # t
  # compare
  t.test(count~donor, data=exp2d[donor=="wt" | donor=="sox10"]) # matches, this is Wel
  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 Welch t-tests.

13.3.5 Adding custom p-values

13.3.6 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.

```r
# 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 com-
puted using the `emmeans` ("estimated marginal means") function from the em-
means package.

```r
# 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
```

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

```r
#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.le
geom_errorbar(data=m1.emm, aes(x=treatment, ymin=asymp.LCL, ymax=asymp.UCL, group=st
                position=pd, width=0.1) +
NULL
```
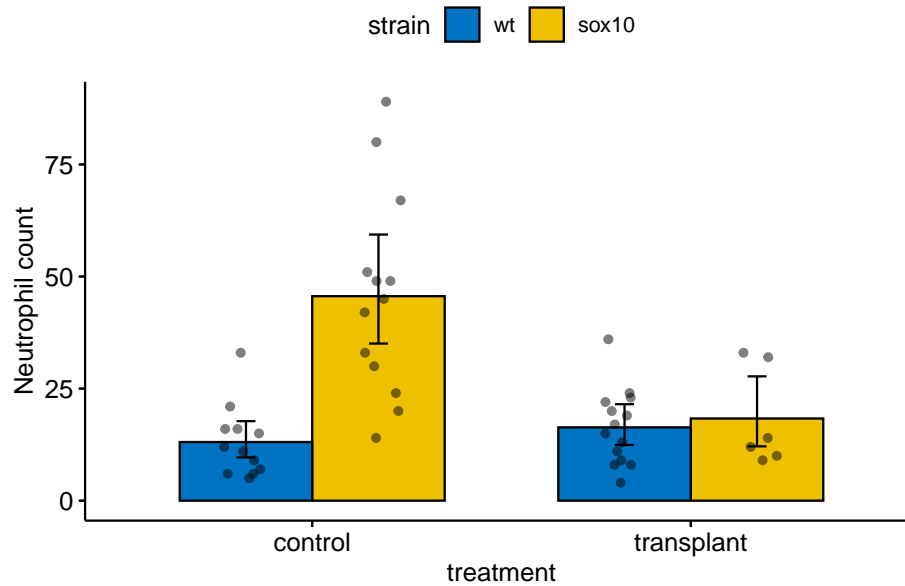

13.3.7 Interaction plot

```
#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=st
  geom_line(position=pd) +
  ylab("Neutrophil count") +
  scale_color_jco() +
  theme_pubr() +
  NULL
```
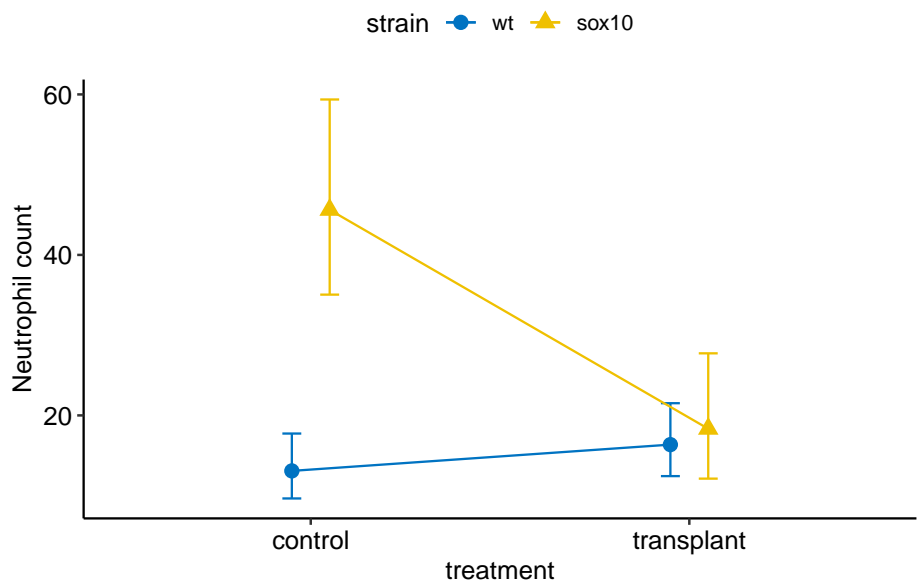
Part IV: More than one X − Multivariable Models

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.
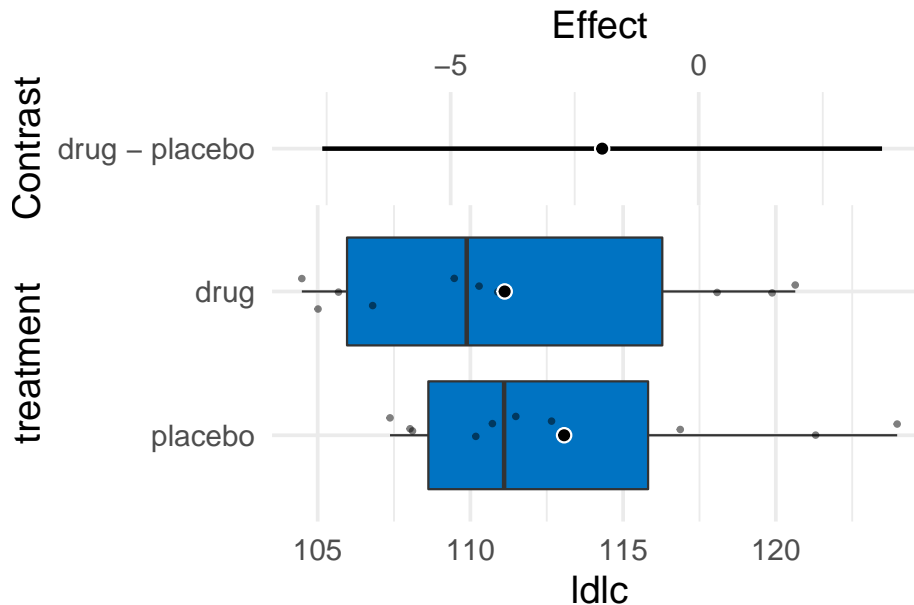
Figure 14.1: The fake LDL-C experiment.

$$ldlc = \beta_0 + \beta_1 treatment + \varepsilon \tag{14.1}$$

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.
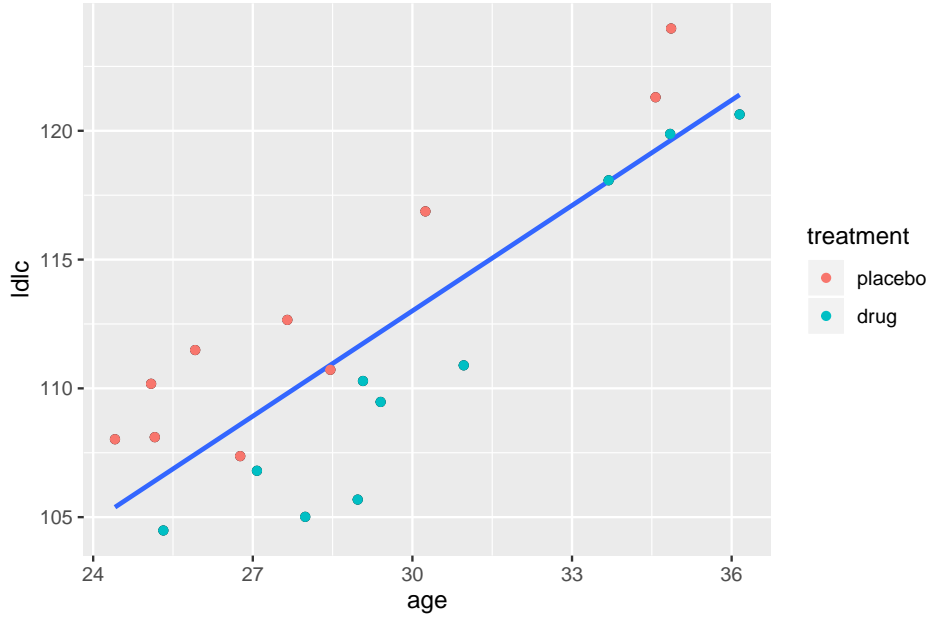
Figure 14.2: Linear regression of *ldlc* on dietary *fat* fit to the fake LDL-C data. The points are color coded by treatment.

$$ldlc = \beta_0 + \beta_1 age + \varepsilon \tag{14.2}$$

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

$$ldlc.r = \beta_0 + \beta_1 treatment + \varepsilon \qquad (14.3)$$

where *ldlc.r* is the set of residuals.
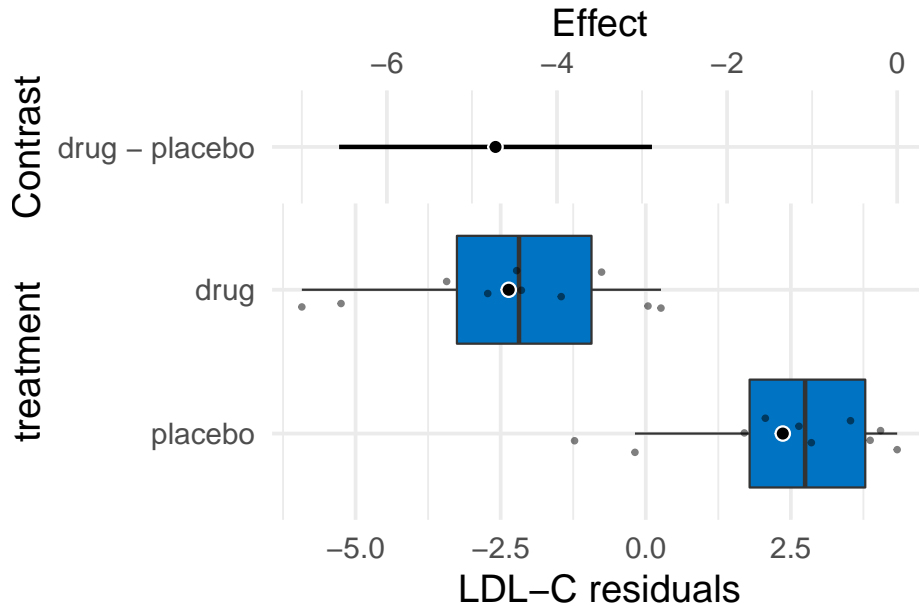
Now the estimate of the effect is -4.7 mg/dL blood and the SE is only 0.88. In this two-stage analysis (stage 1: fit ldlc ~ age to get residuals, stage 2: fit residuals ~ treatment), we have *eliminated the effect of age* on the variance of the response and, as a consequence, the estimate of the effect 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 estmate of the treatment effect by eliminating variance of a covariate (*age*) is to simply add the covariate to the original linear model.

$$ldlc = \beta_0 + \beta_1 age + \beta_2 treatment + \varepsilon \qquad (14.4)$$

which results in the Harrell Plot
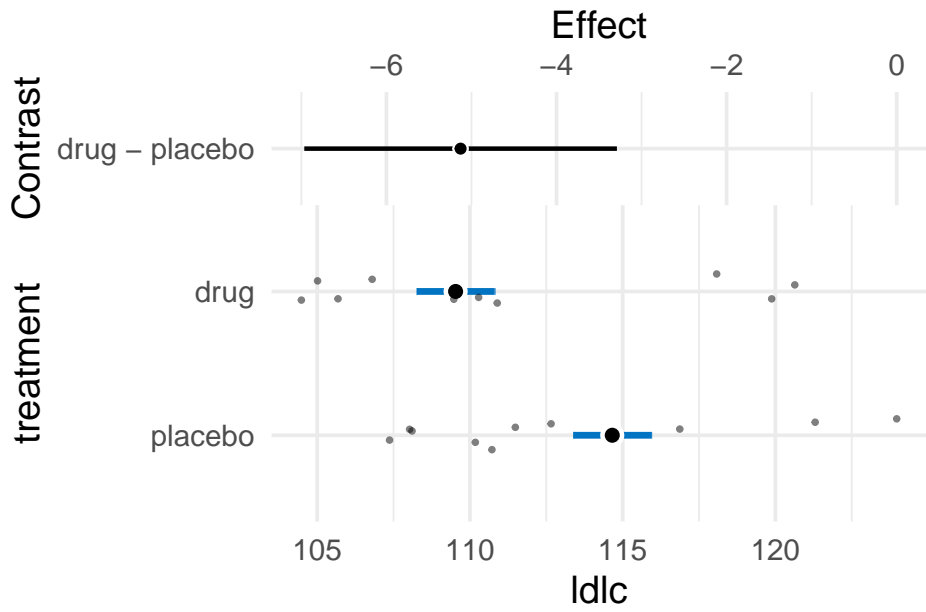
and the coefficient table

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

$$Y_{post} - Y_{pre} = \beta_0 + \beta_1 Treatment + \varepsilon \qquad (14.5)$$

Don't do this. Instead, add the pre-treatment measure into the model as a covariate.

$$Y_{post} = \beta_0 + \beta_1 Y_{pre} + \beta_2 Treatment + \varepsilon \qquad (14.6)$$

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 ($\Delta Y = Ypost - Y_{pre}$) 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).
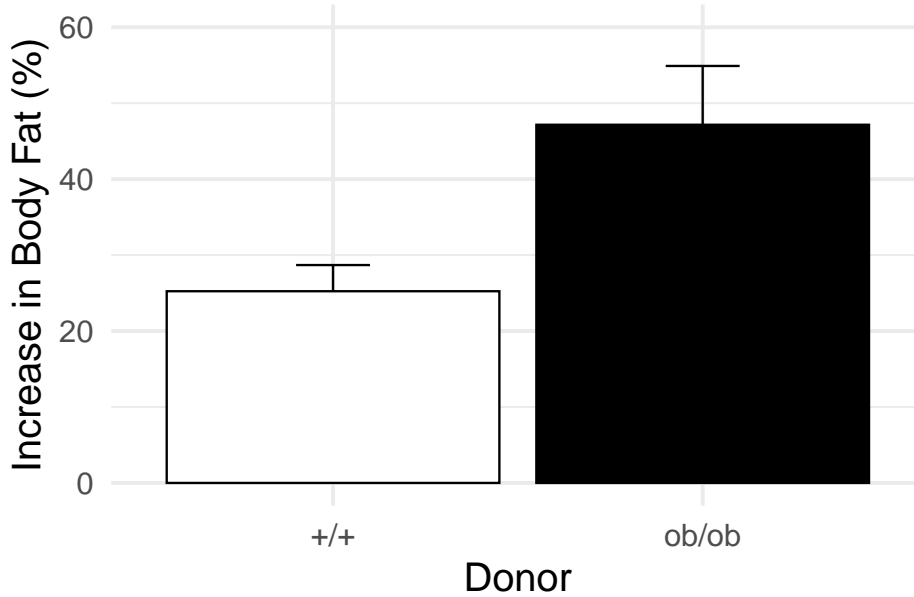
Figure 14.3: Figure 3c of Turnbaugh *et al* 2006. This figure was generated with simulated data matching the summary statistics given in Turnbaugh *et al* 2006

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 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

$$percent\_change\_fat = \beta_0 + \beta_1 obese + \varepsilon \qquad (14.7)$$

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 $\frac{fat_{post} - fat_{pre}}{fat_{pre}} \times$

100, so is a function of the change score $\Delta_{fat} = fat_{post} - fat_{pre}$.

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 correlatioun, 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.
2. $\rho = 1$ – if an individuals'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 nill 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

$$\mathrm{E}(\Delta Y_{ob/ob} - \Delta Y_{+/+}) = 0 \tag{14.8}$$

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 *rho* (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

but the change conditional on baseline is not zero

$$\mathrm{E}(\Delta Y_{ob/ob} - \Delta Y_{+/+}) \neq 0 \qquad (14.9)$$

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 $DeltaY$ 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 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
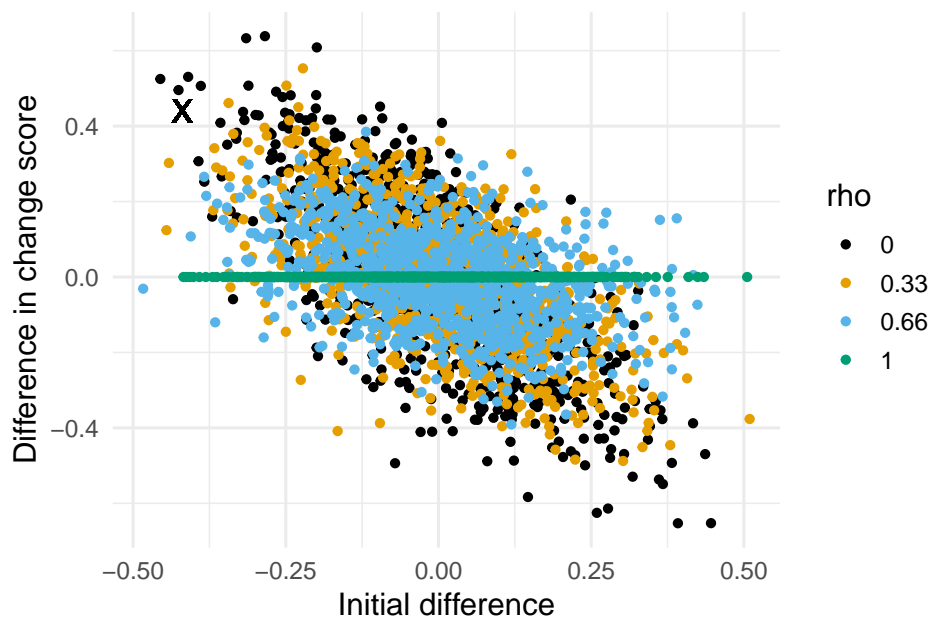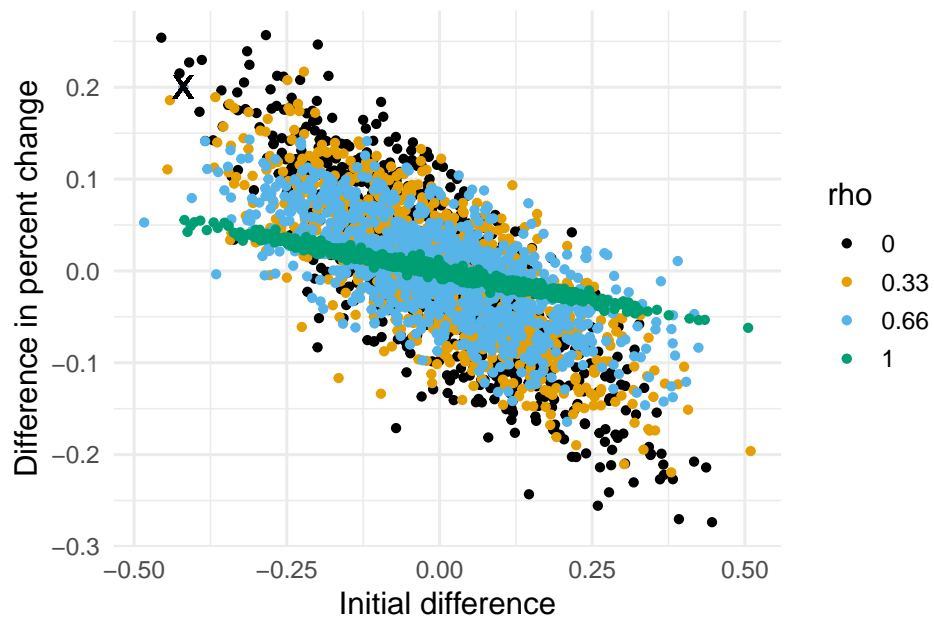
Figure 14.5: Effect of initial difference in weight on the difference in percent change. Increased initial difference in weight results in an increased differences in Percent change between treatment and control. Four different values of *rho* (the correlation between initial and final weights) were simulated. Note there is no value of *rho* where the difference in percent change is independent of the initial difference. The X gives the values in the original Turnbaugh data.

the initial test isn't strictly valid as it does not take into account this decision 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

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 × symbol: $Treatment \times Sex$. Interactions are ubiquitous, although sometimes they are small enough to ignore with little to no loss of understading.

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 µAtm and 1100 µAtm) 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 ($CO2$)

CO2

	400 µatm	1100µATM
13C	n=6	n=6
18C	n=6	n=6

T

The upper left cell represents the combination of 13 C and 400 µAtm 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 µATM higher CO2). And finally, the replicates in the bottom right cell were grown with both the added temperature (+5 C) and added CO2 (+700 µATM). 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

$$Resp = \beta_0 + \beta_1 Temp^+ + \beta_2 CO2^+ + \beta_3 Temp^+ CO2^+ \qquad (15.1)$$

Again, $Temp^+$ and $CO2^+$ are dummy variables. The model also includes $Temp^+ CO2^+$, 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 $CO2^+$), which can be verified with the model matrix (which here, is computed from the subset of the data that includes 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

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 ($b_3 = \bar{Y}_{Temp+CO2+} - (b0 + b1 + b2)$). An interaction is a **non-additive effect**. Think about this. Adding 5 C increases respiration by 4.51 units. Adding 700 µATM CO2 decreases respiration by .32 units. If these effects were purely additive, then adding both 5 C and 700 µATM 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.
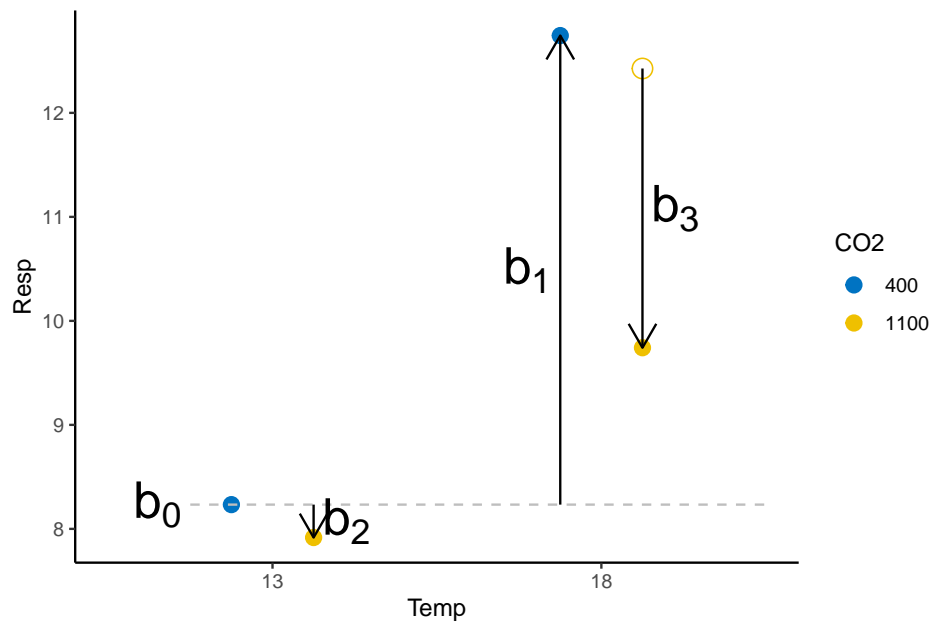
Figure 15.1: Meaning of coefficients in factorial model. b0 (dashed line) is the mean of the reference. b1 (length of vector b1) is the mean of the Temp treatment minus the mean of the reference. b2 (length of vector b2) is the mean of the CO2 treatment minus the mean of the reference. b3 (length of vector b3) 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

$Pr(>|t|)$

(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

emmean

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

emmean

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 CO_2 is 400 µAtm and an effect when CO_2 is 1100 µAtm. Similarly, there is an effect of the 1100 level of CO_2 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 CO_2 at each level of Temp. The contrasts in the bottom row are the simple effects of Temp at each level of CO_2. 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

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 µAtm. The 3rd line is the effect of the 1100 µAtm 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 µAtm

1100 µAtm

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.

$$Resp = \beta_0 + \beta_1 Temp^+ + \beta_2 CO2^+ \tag{15.2}$$

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

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 heterogenity 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

$$Y = \beta_0 + \beta_1 A^+ + \beta_1 B^+ + \beta_3 C^+ + \beta_4 A^+ B^+ + \beta_5 A^+ C^+ + \beta_6 B^+ C^+ + \beta_7 A^+ B^+ C^+ + \varepsilon \tag{15.3}$$

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 ~ 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 ~ 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 ~ 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)

```r
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`.

```r
m1.effects <- summary(contrast(m1.emm,
                 method="revpairwise",
                 adjust="none",
                 simple = "each",
                 combine=TRUE),
          infer=c(TRUE,TRUE))
m1.effects
```

```
##  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

```r
m1.emm.1 <- emmeans(m1, specs=c("Temp"))
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
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"))
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
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
```
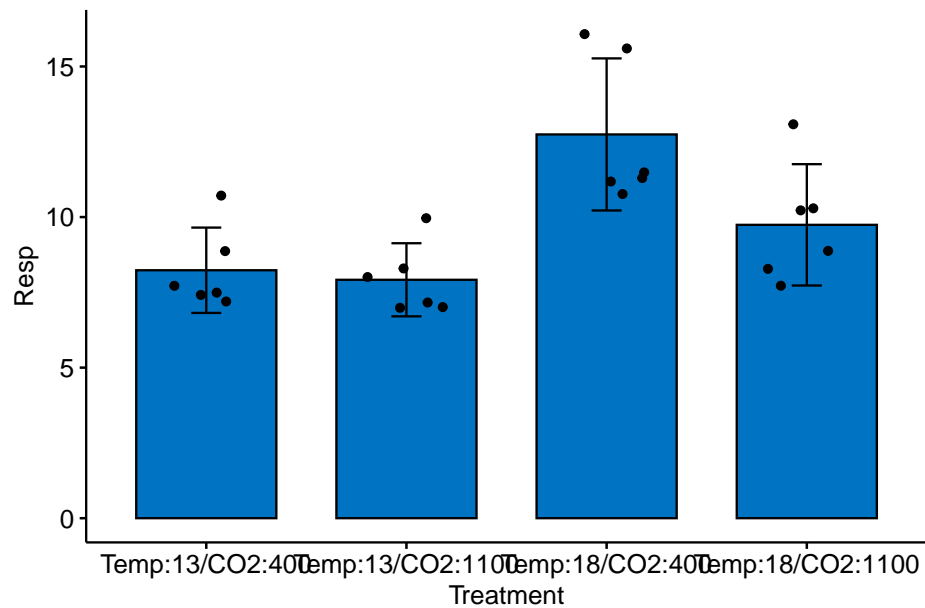

15.3.7.2 Plots that communicate the factorial design

```
# 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=CO2),
        color="black",
        position=position_jitterdodge(jitter.width=0.2),
        show.legend=FALSE,
        alpha=0.5) +
  scale_fill_jco() +
  NULL
```

```r
# "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=emmean,
                          shape=CO2,
                          color=CO2,
                          group=CO2)) +
  geom_point(position=pd, size=3) +
  geom_errorbar(aes(x=Temp,
                                ymin=lower.CL,
                                ymax=upper.CL,
                                group=CO2)
            , 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=CO2),
              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=emmean, group=CO2),
              color="red",
              position=pd) +
  geom_line(data=m1.emm.dt,
              aes(x=Temp, y=emmean, 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. 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)
```
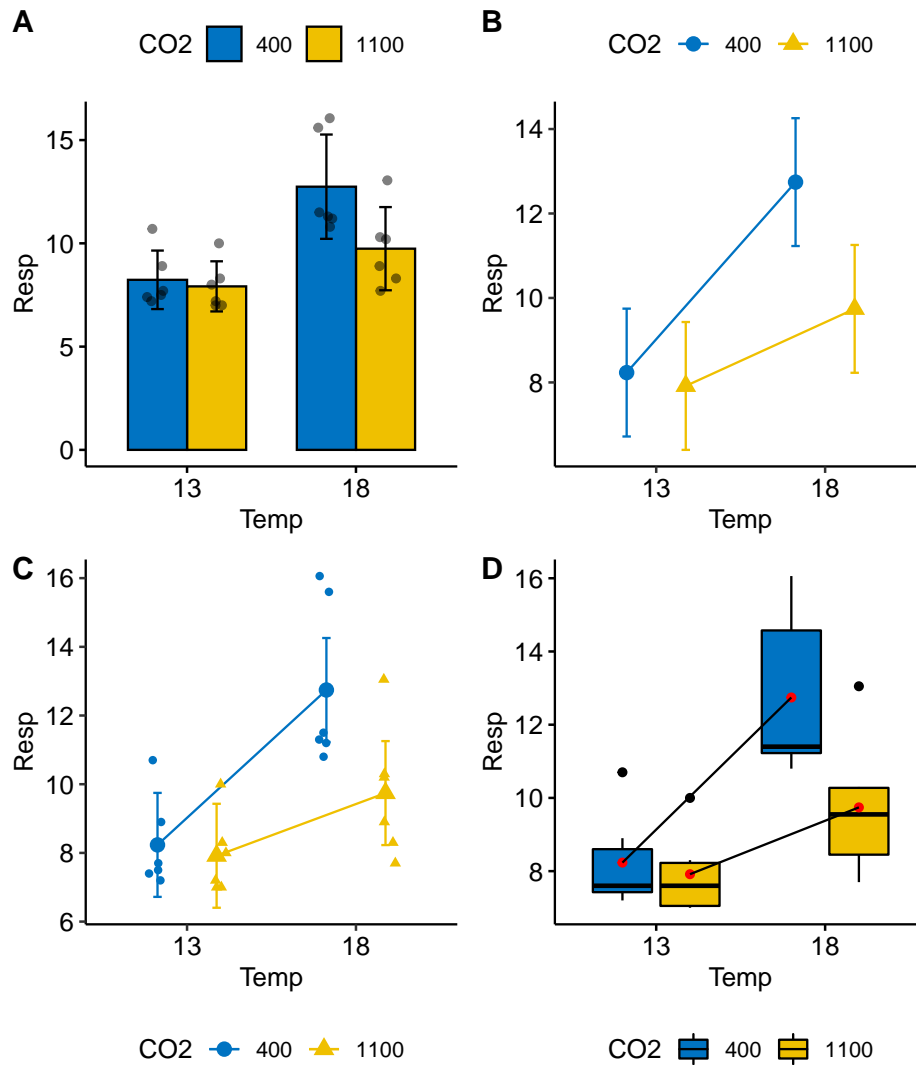
Figure 15.2: Interaction plots. (B) is the classic interaction plot, which is characterized by lines connecting the groups that share the same Factor B level. This line allows one to visual the effect of Factor A (the slope) at each level of Factor B.

```r
# 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
```
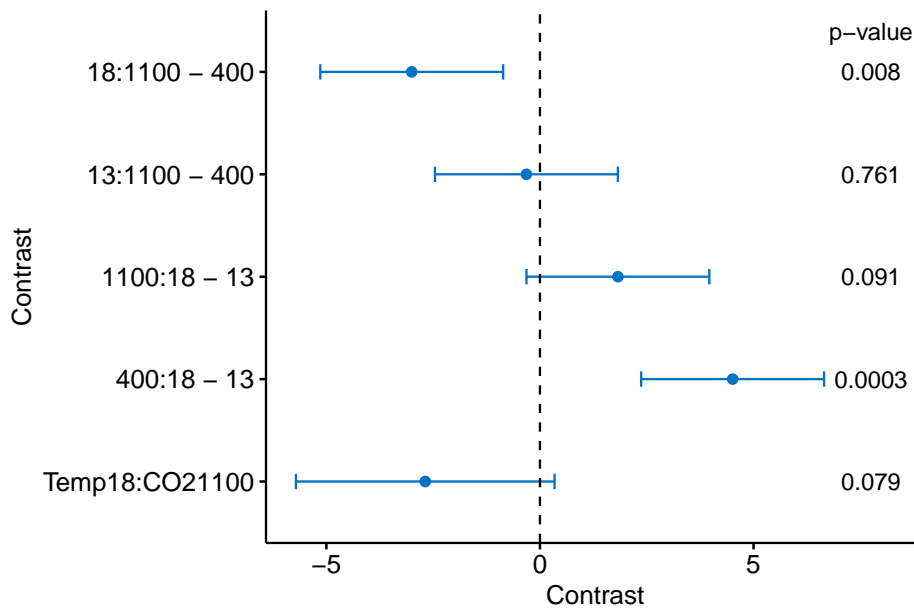
This effects plot shows the four simple effects, the single interaction (Temp18:CO21100), and their 95% confidence intervals. In the original paper, the researchers were testing a scientific (not statistical!) model that predicted no interaction between CO2 and Temp, and the researchers argued that these 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 course (or the signal:noise ratio is to 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))
```
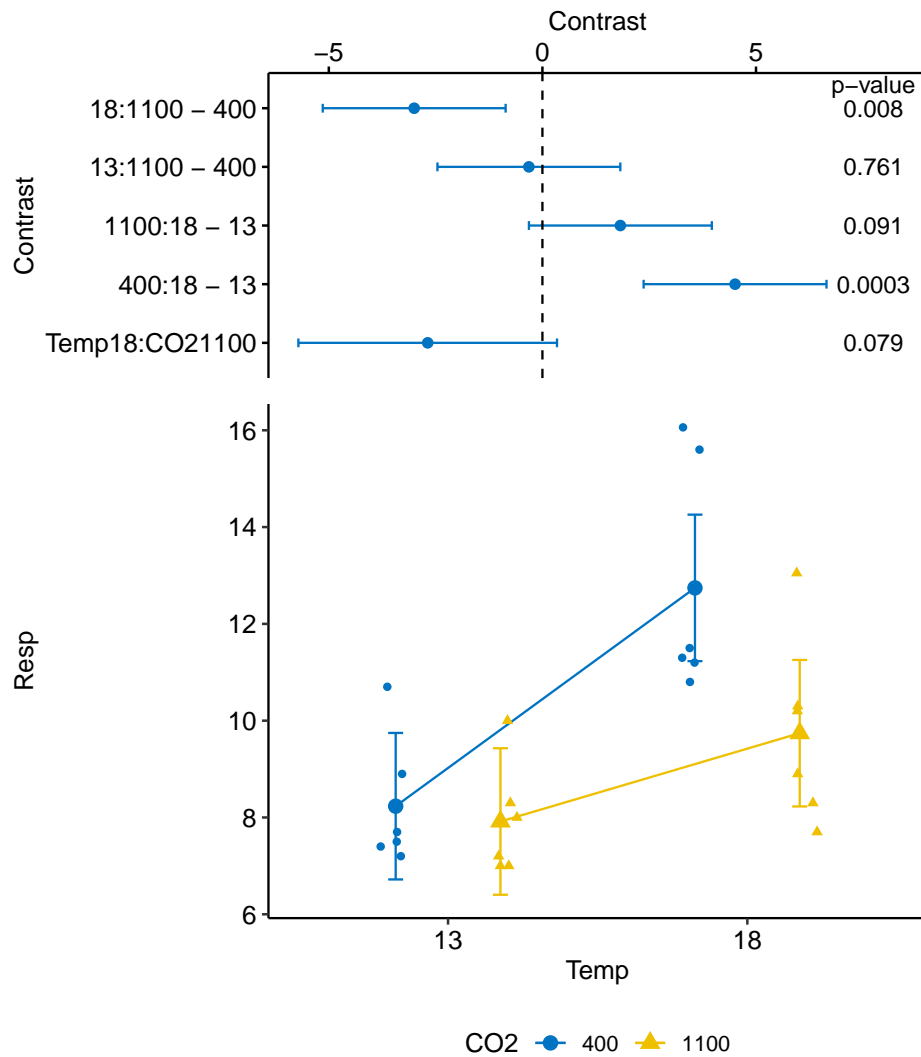
The effects and interaction plot are combined into a single plot.

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 atomospheric 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 soild 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
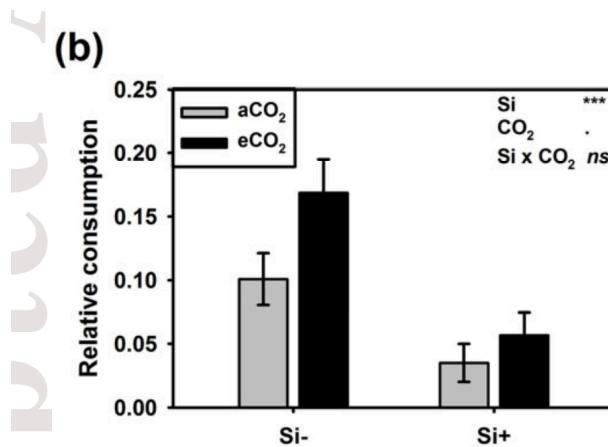
Figure 4. (a) Effects of silicon treatments on the relative growth rate [=mass gained (g)/time(days)] and **(b)** relative consumption of roots [=food ingested (mg change in dry mass)/ initial body mass (mg fresh mass)] under aCO_2 and eCO_2. Levels of significance are shown for effects of silicon and CO_2 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
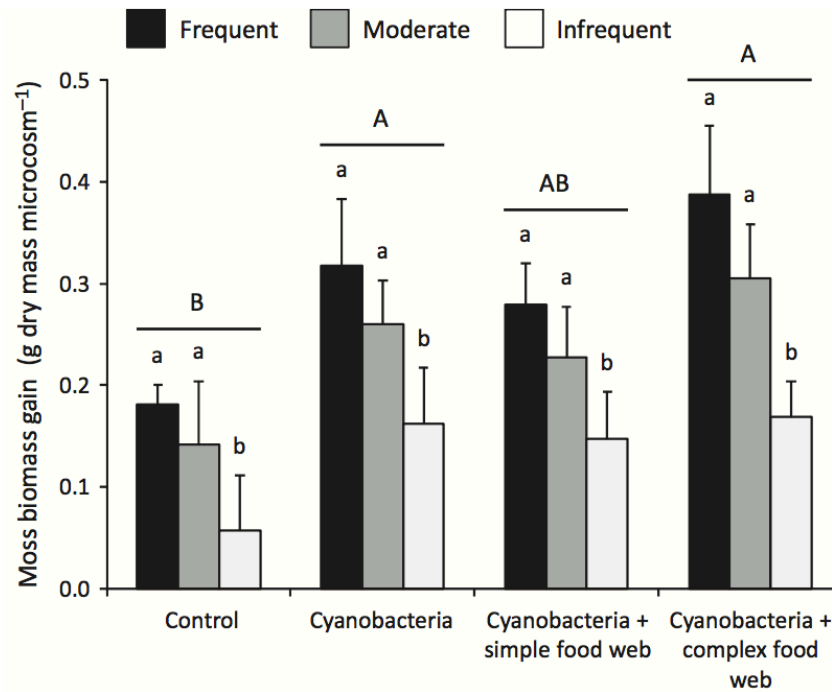
Figure 3 Responses of moss biomass gain to food-web structure (control consisting of sterilised moss with no cyanobacteria added, cyanobacteria, cyanobacteria + simple food web, cyanobacteria + complex food web) and precipitation regime (frequent, moderate, infrequent). Uppercase letters above groups 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 analyis 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 re-fitting 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. Regardles 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 hypotheis 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

$$Resp = \beta_0 + \beta_1 Temp + \beta_2 CO2 + \beta_3 TempCO2 + \varepsilon \qquad (16.1)$$

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 kth level mean has no freedom to vary; its value has to be $k\bar{\bar{Y}} - \sum_i^{k-1} Y_i$. 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 dived 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

$$b_1 = \overline{\overline{Resp}} - \text{E}(Resp|Temp^+) \tag{16.2}$$

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

$$b_2 = \overline{\overline{Resp}} - \text{E}(Resp|CO2^+) \tag{16.3}$$

3. Temp:CO2 – $p = \text{prob}(F \geq F_o | Temp, CO2)$. 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.

$$b_3 = \mathrm{E}(Resp|Temp^+, CO2^+) - (\overline{\overline{Resp}} - b_1 - b_2) \qquad (16.4)$$

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

$\Pr(>|t|)$

(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 not 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

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 $Temp$ – $CO2$ 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 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)
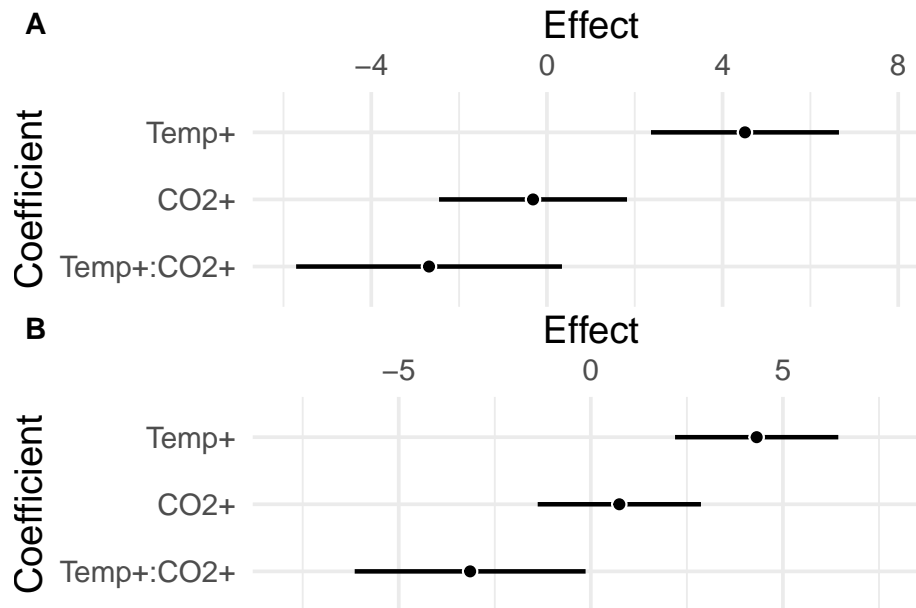
Figure 16.1: Forest plots (the upper part of a Harrell plot) of the actual and fake urchin data. A) Real urchin data. The interaction effect is not significant ($p = 0.079$). B) Fake urchin data. The interaction effect is significant ($p = 0.04$).

the 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 interactioni 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 denomintor 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

	Df	Sum Sq	Mean Sq	F value	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 $CO2$ and half experienced high $CO2$

2. The hypothesis tested by the p-value for $CO2$ is conditional on $Temp$. That is, the difference in metabolism between $CO2+$ and $CO2-$ 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 CO2 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.

```r
cn_rows <- which(urchin[, Temp]=="Temp-" & urchin[, CO2]=="CO2-") # gives the rows of
urchin_unbalanced <- urchin[-cn_rows[1],] # deletes the row that is in first element o
urchin.t1 <- lm(Resp ~ Temp*CO2, data=urchin_unbalanced)
anova(urchin.t1)
```

```
## Analysis of Variance Table
##
## Response: Resp
##           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(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 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

```r
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 continous and categorical. The classical analysis of multiple X variables is **multiple regression**, sometimes called **multivariable regression** and occassionally, but incorrectly, called **multivariate regression** – "multivariate" refers to multiple Y variables.

The models in this chapter have the structure

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + ...\beta_p X_p + \varepsilon \tag{17.1}$$

% 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

$$R^2 < -1 - \frac{SS_{residual}}{SS_{total}} \tag{17.2}$$

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 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 correlewhich 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 V: Expanding the Linear Model – Generalized Linear Models and Multilevel (Linear Mixed) Models

Chapter 18

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 18.1 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.
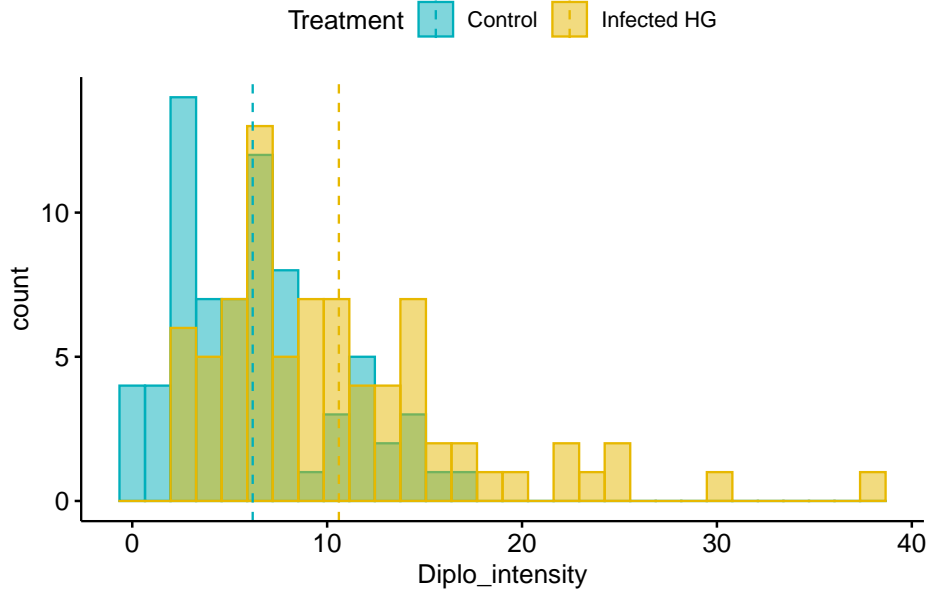
Figure 18.1: Histogram of the count of a trematode parasite larvae in Control vs. Infected fish. Fish in the Infected treatment are infected with a tapeworm.

18.1   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

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{18.1}$$

$$\varepsilon \sim N(0, \sigma) \tag{18.2}$$

That is, we can think of a response as the sum of some systematic part and "random errror", 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

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{18.3}$$
$$\varepsilon \sim Poisson(\lambda) \tag{18.4}$$

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 (18.4) 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 (18.4) 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

$$y_i \sim N(\mu_i, \sigma) \tag{18.5}$$
$$\mathrm{E}(Y|X) = \mu \tag{18.6}$$
$$\mu_i = \beta_0 + \beta_1 x_i \tag{18.7}$$

That is, a response is a random draw from a normal distribution with mean *mu* (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")

$$y_i \sim \mathrm{Prob}(\mu_i) \tag{18.8}$$

2. **the systematic part**, which is a linear predictor (I like to think about this as the deterministic part)

$$\eta = \mathbf{X}\boldsymbol{\beta} \tag{18.9}$$

3. a **link function** connecting the two parts

$$\eta_i = g(\mu_i) \tag{18.10}$$

μ (the Greek symbol mu) is the conditional mean (or expectation $\mathrm{E}(Y|X)$) 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 sale is $\exp(\hat{\eta})$ and a coefficient on the response scale is $\exp(b_j)$.

18.2   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 18.1 for the control and Infected HG treatment levels.

---

[1]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

18.2.1 Modeling strategy

NHST blues – Students are often encouraged by textbooks, colleagues, or the literature to start the analysis by first "testing" assumptions with hypothesis tests – for example using a Shaprio-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 increses, 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 (18.2)) because the diagnostic plots for model checking a linear model use the residuals of the fit model.

$$Diplo\_intensity_i = \beta_0 + \beta_1 Uninfected_i + \beta_2 Infected\_LG_i + \beta_3 Infected\_HG_i + \varepsilon_i \tag{18.11}$$

$$\varepsilon \sim N(0, \sigma) \tag{18.12}$$

18.2.2 Checking the model I – a Normal Q-Q plot

Figure 18.2A 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 18.2B, 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
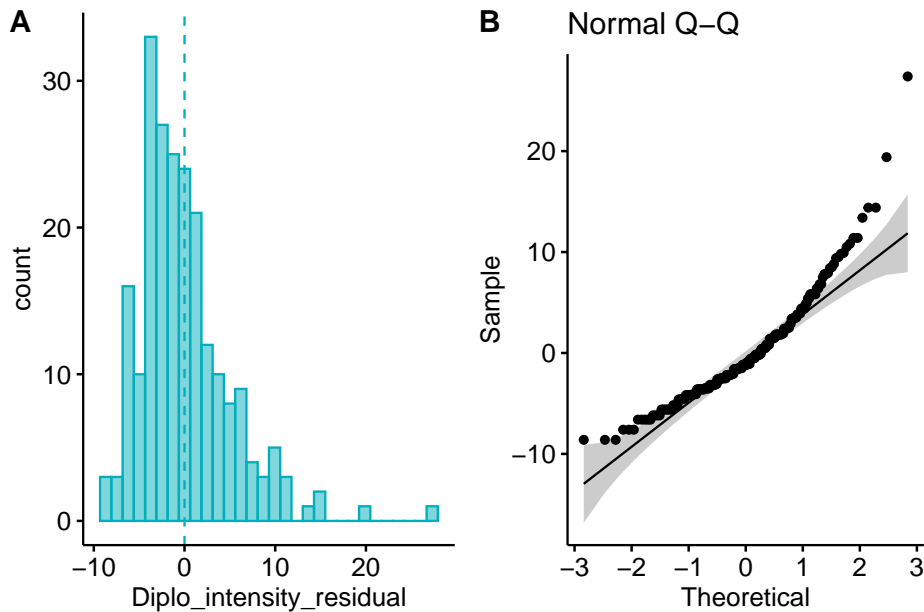
Figure 18.2: Diagnostic plots of stickleback parasite data. A) Distribution of the residuals of the fit linear model. B) Normal Q-Q plot of the residuals of the fit linear model.

than the theoretical values. Thinking about this as counts, this means that are 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.

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 **stan-**

dardized 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 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).
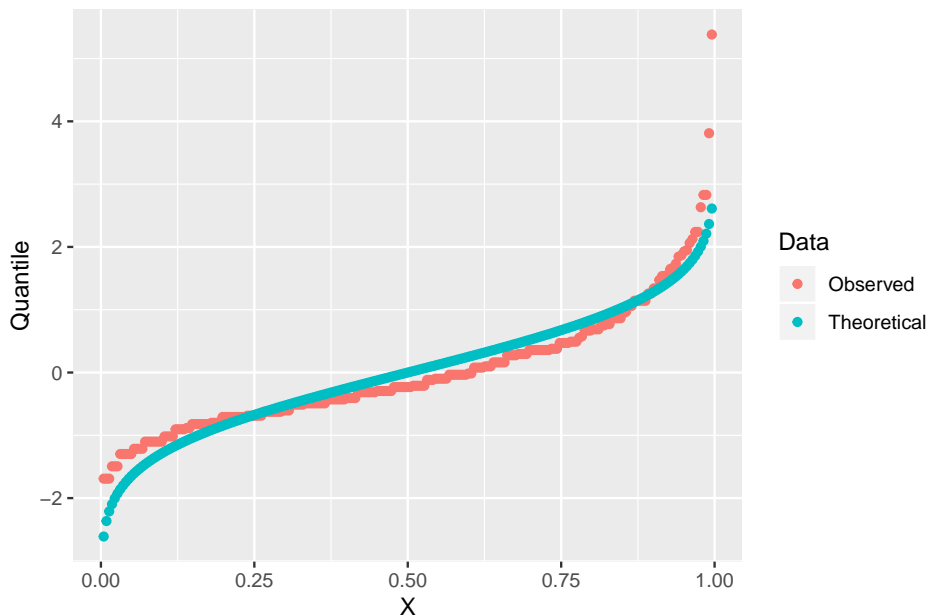
18.2.3   Checking the model II – scale-location plot for checking homoskedasticity
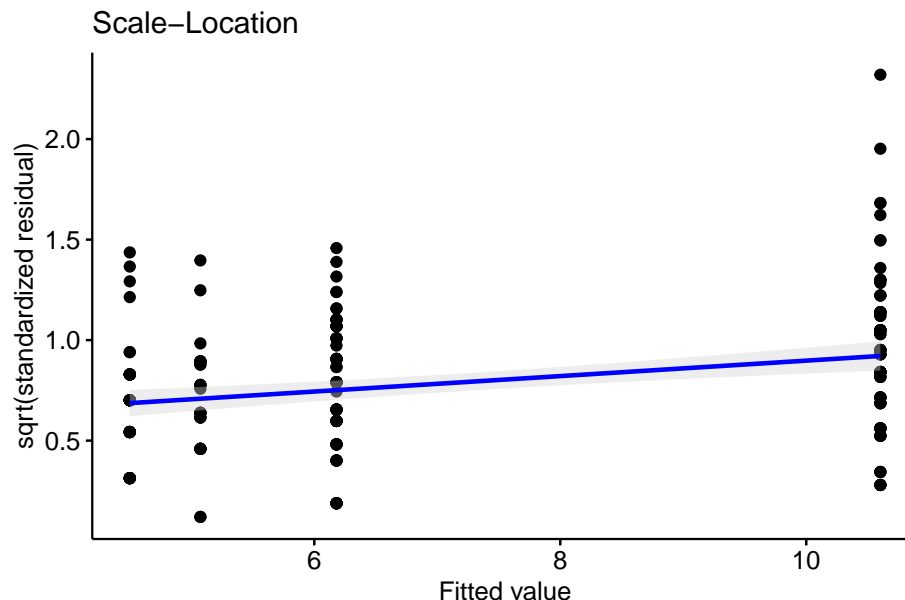
Scale–Location

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 values[2]. 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.

18.2.4   Two distributions for count data – Poisson and Negative Binomial

The pattern in the normal Q-Q plot in Figure 18.2B 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

---

[2]fitted values are the predicted values, $\hat{Y}$

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.
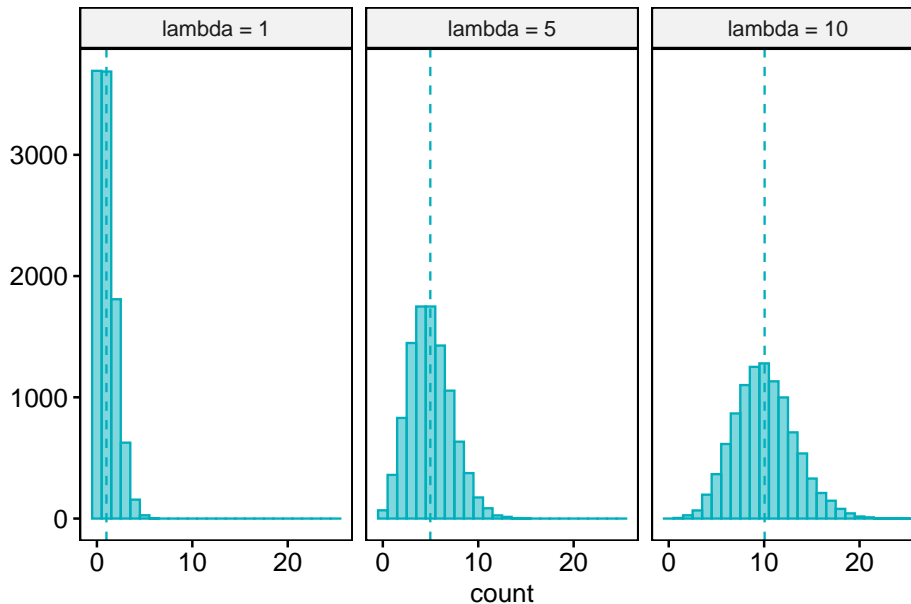
2. Negative Binomial – The negative binomial distribution is a discrete prob-

---

[3]the variance is less than that expected by the probability model

[4]the variance is greater than that expected by the probability model

ability 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 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.

18.2.5   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

$$Diplo\_intensity_i \sim Poisson(\mu_i) \tag{18.13}$$
$$\mathrm{E}(Diplo\_intensity|Treatment) = \mu \tag{18.14}$$
$$\mu_i = \exp(\eta_i) \tag{18.15}$$
$$\eta_i = \beta_0 + \beta_1 Uninfected_i + \beta_2 Infected\_LG_i + \beta_3 Infected\_HG_i$$
$$\tag{18.16}$$

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 $X = x_i$.

18.2.6   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 more zeros than expected by the theoretical distribution. If so, the observed distribution is **zero-inflated**

18.2.6.1 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 distribitued 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 defintion 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.

Quantile Residual Q–Q Plot
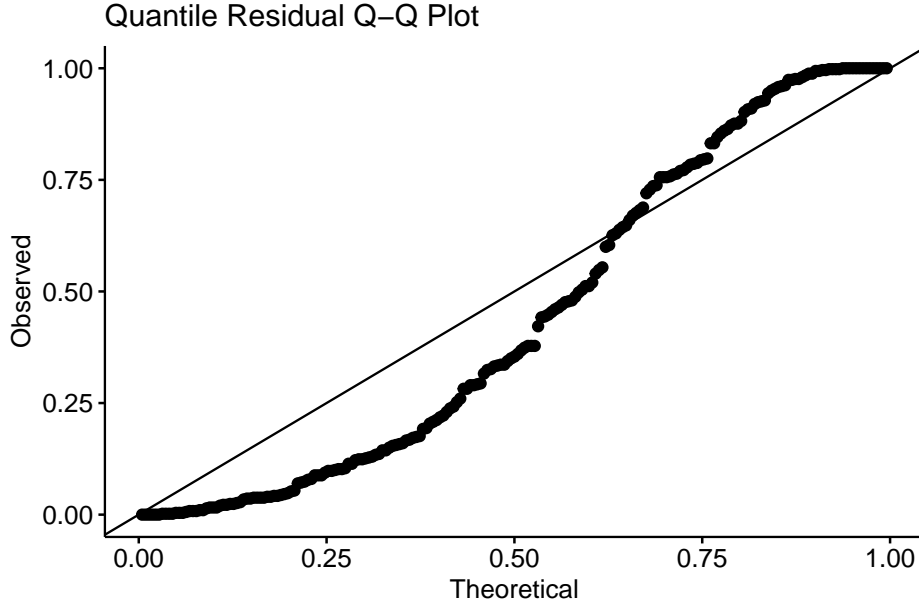
The Q-Q plot using quantile residuals with a Poisson distribution indicates that the counts of *Diplostomum* larvae in the eyes of the threespine stickleback are not well approximated by a Poisson distribution – there are too many observed values near the ends of the expected tails, indicating the expected values are not spread out enough. This pattern emerges because the observed counts are overdispersed compared to a Poisson distribution.

18.2.6.2   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

$$\frac{\sum r_i}{df} \tag{18.17}$$

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.
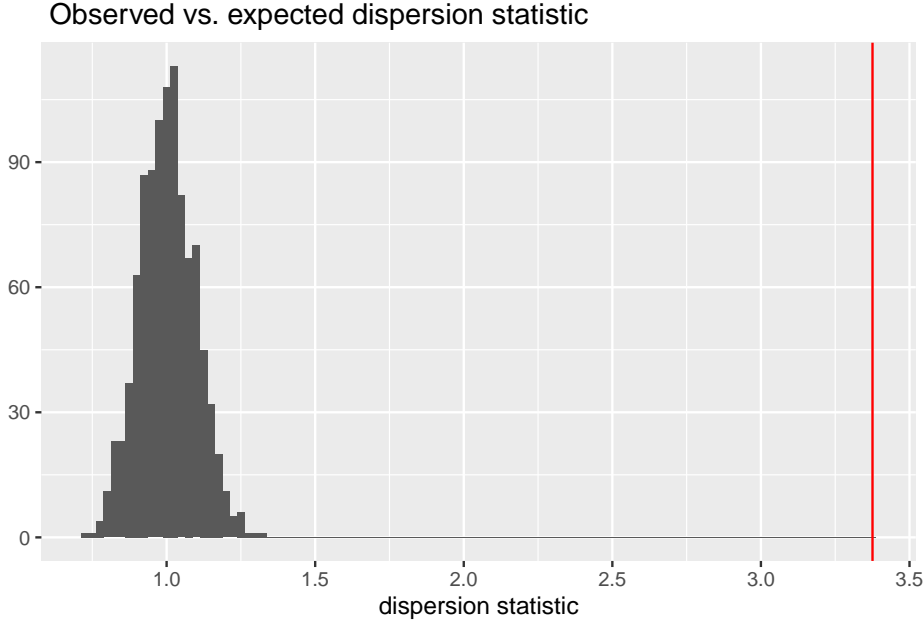3. Repeat 1 and 2 N_{iter} times.

Figure 18.3: Observed vs. expected dispersion statistic. The observed statistic marked by the red line. The histogram of expected statistics are from 1000 simulations of the observed data.

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.

18.2.7 Fitting a GLM with a Negative Binomial distribution to the worm data

The model is
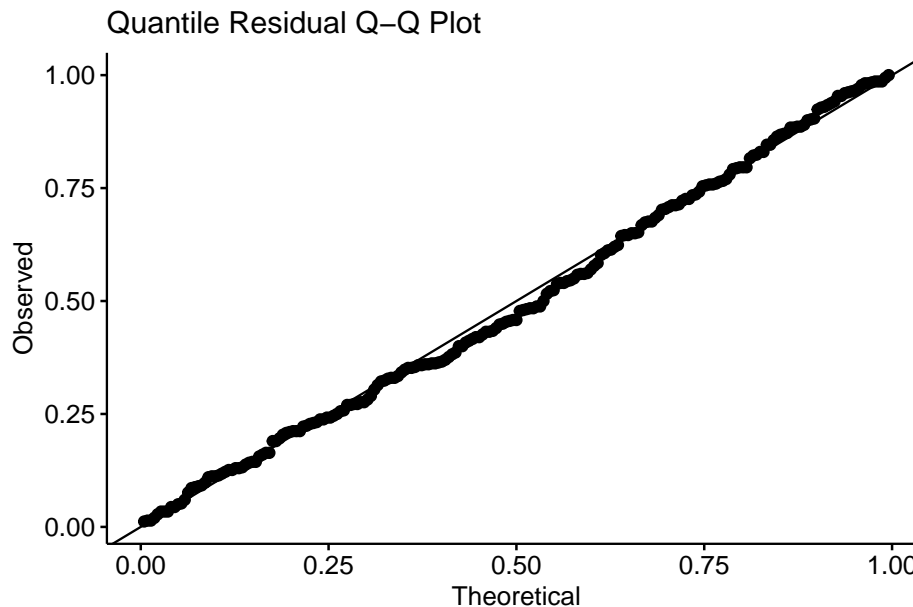
$$Diplo\_intensity \sim NB(\mu, k) \tag{18.18}$$
$$\mathrm{E}(Diplo\_intensity | Treatment) = \mu \tag{18.19}$$
$$\mu = \exp(\eta) \tag{18.20}$$
$$\eta = \beta_0 + \beta_1 Uninfected + \beta_2 Infected\_LG + \beta_3 Infected\_HG \tag{18.21}$$

This model specifies a negative binomial distribution but otherwise is just like that above specifying a Poisson distribution.

18.2.7.1   Model checking

Quantile Residual Q–Q Plot

A quantile residual Q-Q plot of the GLM model fit with negative binomial distribution is illustrated above. This looks pretty good.

18.2.7.2   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
```

While the means on response scale are

```
## Treatment    response     SE  df asymp.LCL asymp.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 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
```

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

$$RR_{Infected\_HG} = \frac{\overline{Infected\_HG}}{\overline{Control}} \tag{18.22}$$

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 $\overline{Control} * RR_{Infected\_HG}$ (remember that with a linear model the mean would be $b_{Control} + b_{Infected\_HG}$). Check and see if this works.

18.3   Working in R

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)
```

DHARMa scaled residual plots
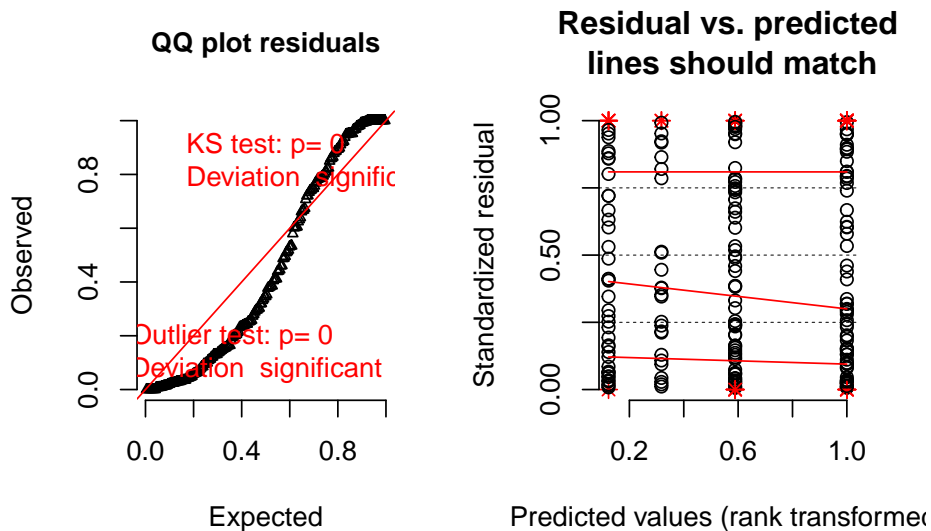
A plot of the dispersion statistic can be generated using the object returned by the `SimulateOutput` function but with `refit = TRUE`, which refits a model each iteration. This refitting isn't necessary if only the quantiile residuals are needed. The Dharma package does not divide the sum of squared 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
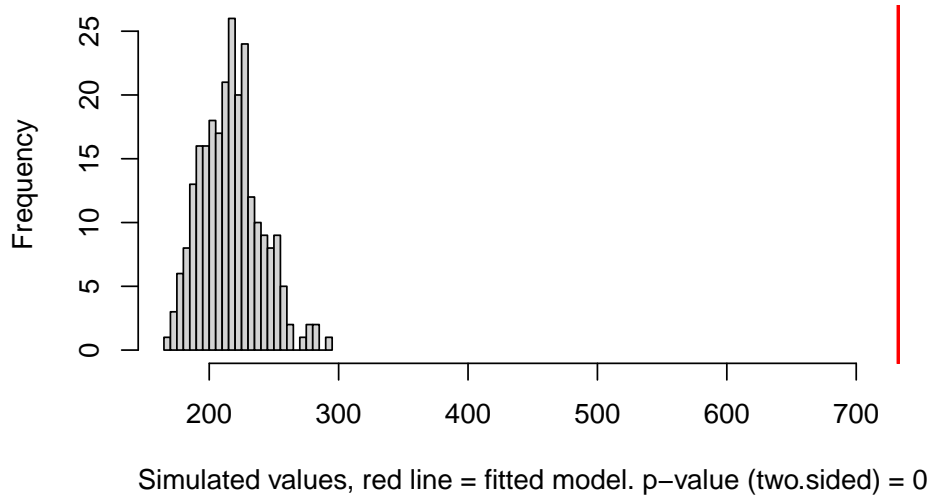
```
##
##  DHARMa nonparametric dispersion test via mean deviance residual
##  fitted vs. simulated-refitted
##
## data:  simulationOutput
## dispersion = 3.3788, p-value < 2.2e-16
## alternative hypothesis: two.sided
```

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

QQ plot residuals

KS test: p= 0.789
Deviation n.s.

Outlier test: p= 1
Deviation n.s.

Observed

0.0   0.4   0.8
Expected

**Residual vs. predicted
lines should match**

Standardized residual

0.00   0.50   1.00

0.2   0.6   1.0
Predicted values (rank transformed)
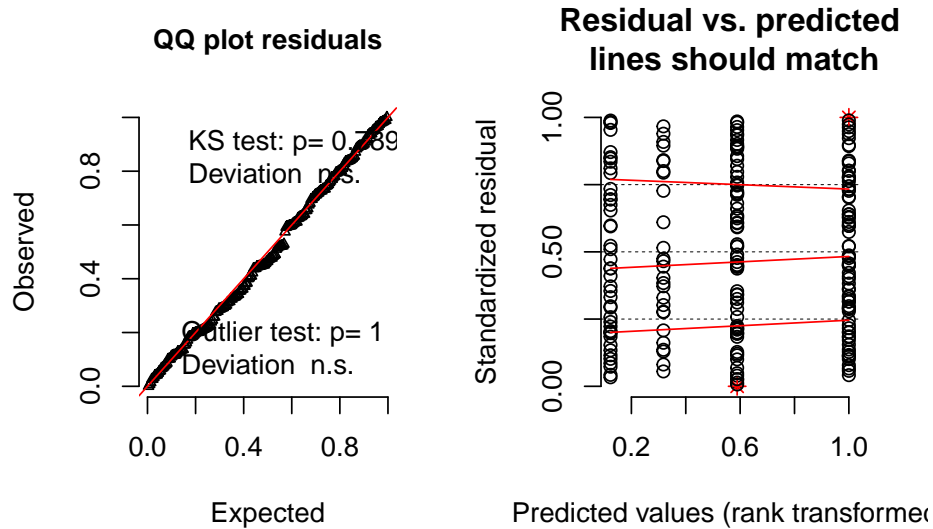
```r
# link scale
emm <- emmeans(fit.nb, specs="Treatment")
emm
```

```
##  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
```

```r
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
```

```r
# response scale
emm.response <- emmeans(fit.nb, specs="Treatment", type="response")
summary(contrast(emm, method="trt.vs.ctrl", adjust="none", type="response"), infer=c(TI
```

```
##  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
```

18.4   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 19

Linear mixed models

19.1 Random effects

Researchers often collect data in batches, for example

1. Researchers 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 all seedlings. 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.

---

[1] Giffard, 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. A researcher is using a mouse model to compare growth of a wildtype and engineered mutant strain of *Staphylococcus*. A small spot on both right and left forelimbs of ten mice is shaved and abraded. The two strains are randomly assigned to a side (so each mouse is infected with each strain). Small, uncontrolled, environmental factors (including immune responses) will differ between the mice but be common between the two limbs within a mouse and we would expect a common response to this uncontrolled variation on top of the differential response to each treatment. As a consequence, the two measures of growth within a mouse are not independent.

4. 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 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.

5. An ecologist wants to measure the effect of habitat on chick growth in a bird. Five individuals nest in artifical nest boxes built on the boundary between the forest and a large, agricultural field. Five other individuals nest in boxes built deep in the interior of the forest. Chicks in each nest are weighed 13 days after hatching. Small, uncontrolled, environmental factors (including parenting, food availablity, temperature, etc.) will differ between the nests but be common within the nests and we would expect a common response to this uncontrolled variation on top of the differential response to each treatment. As a consequence, the measures of weight within a nest are not independent.

6. A physiologists 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.

7. 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.

The batches – plots in experiment 1, individuals in experiment 2, mice in experiment 3, plots in experiment 4, nests in experiment 5, cultures in experiment 6, and tanks in experiment 7 – are often referred to as **blocks** or **clusters**. I'll generally use the term "block" in this book. The blocks are the experimental units, meaning that it is at this level that the experimenter is controlling the conditions. There is systematic variation at two levels: among treatments due to treatment effects and among blocks due to **block effects**. This among-block variation is the **random effect**. An assumption of modeling random effects is that the blocks (plots/individuals/mice/nests/cultures/tanks) are a random sample of the blocks that could have been sampled. This is often not strictly true as blocks are often **convenience samples**.

The multiple measures within a block are often called **repeated measures**, especially if the block is an experimental animal such as a mouse or human. If multiple measures within a treatment level within a block (that is, within a $block \times treatment$ combination) are taken over time, the data are **longitudinal**. Sometimes in cell biology, the subsamples within a treatment within a block are called "replicates", as they are replicates of this $block \times treatment$ combination, but this can be confusing because the treatments are replicated at the level of the block and not at the level of the subsamples within a treatment by block combination. The blocks are the independent experimental units. Instead the multiple measures of the response within a $block \times treatment$ combination are **subsamples**.

Experiments 1 and 2 are examples of a **complete randomized block with subsampling** design. "Complete" means that each block has all treatment levels or combinations of levels if there is more than one factor. Experiments 3 and 4 are examples of a **complete randomized block** design. The blocks are complete but there is only one measure of the response per treatment. Experiments 5, 6, and 7 are examples of an **incomplete randomized blocks** design. The blocks are incomplete because they do not contain less than all treatment levels and combinations. In these examples, each block contains only one treatment level.

19.2   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 nuissance factors that add additional variance to the response, with the consequence that estimates of treatment effects are less precise, unless the varriance 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

$$y_{ij} = (\beta_0 + \beta_{0j}) + (\beta_1 + \beta_{1j})x_i + \varepsilon_i \tag{19.1}$$

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 random intercept 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.

```
## boundary (singular) fit: see ?isSingular
```

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 components. Another useful way to specify this model is to think about it hierarchically, using

$$y_{ij} = \beta_{0j} + \beta_{1j}x_i + \varepsilon_i \tag{19.2}$$

$$\varepsilon_i \sim N(0, \sigma) \tag{19.3}$$

$$\beta_{0j} = \beta_0 + N(0, \sigma_0) \tag{19.4}$$

$$\beta_{1j} = \beta_1 + N(0, \sigma_1) \tag{19.5}$$

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.

$$y_{ij} \sim N(\mu_{ij}, \sigma) \tag{19.6}$$

$$\mu_{ij} = \beta_{0j} + \beta_{1j}x_i \tag{19.7}$$

$$\beta_{0j} \sim N(\beta_0, \sigma_0) \tag{19.8}$$

$$\beta_{1j} \sim N(\beta_1, \sigma_1) \tag{19.9}$$

19.3   Linear mixed models are flexible

The linear mixed model in Equation (19.1) specifies both a random intercept and a random slope but a researcher might limit the random effect to the intercept only, or less commonly, the slope only. Excluding the random slope from Equation (19.1) results in the model

$$y_{ij} = (\beta_0 + \beta_{0j}) + \beta_1 x_i + \varepsilon_i \tag{19.10}$$

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.
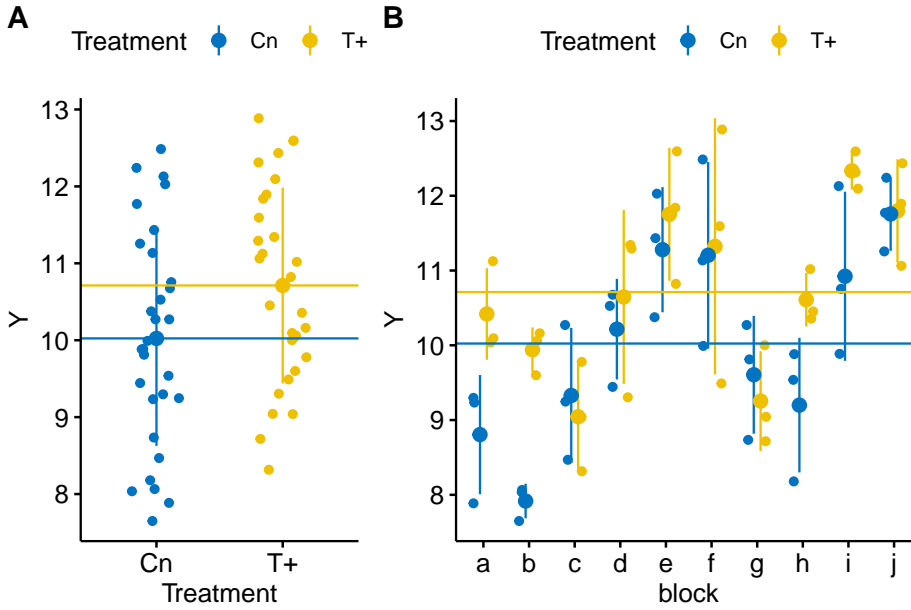
Figure 19.1: Visualizing random effects. A) The response in the two treatment levels. B) The same data but separated by block. The blue line is at the control mean and the yellow line is at the treated mean. The black dots are the mean response within a block.

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 **??**.

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Zu} + \boldsymbol{\varepsilon} \qquad (19.11)$$

where $\mathbf{y}$ is the vector of the response, $\mathbf{X}\boldsymbol{\beta}$ is the linear predictor of fixed effects and $\mathbf{Zu}$ 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

19.4 Visualizing block effects

To visualize random effects due to block, Let's create fake data that look something like experiments 1 or 2, 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 19.1A 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 19.1B 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, c, g, and h, all or most of the responses are below their treatment mean (for example in block a, all the blue points are below the blue line and 2 of 3 yellow points are below the yellow line). In blocks, e, f, i, and j, all or most of the responses are above their treatment mean (for example, in block i, all three yellow points are above the yellow line and 2 of 3 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**.

19.5   Linear mixed models can increase 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 consquence, this block-specific variance becomes part of the error variance. One way to visualize this is by moving the random intercept and random slope components of equation (19.1) to the right and combining it with the observation-specific error

$$y_{ij} = \beta_0 + \beta_1 x_i + (\beta_{0j} + \beta_{1j} + \varepsilon_i) \tag{19.12}$$

% which shows that the random effects β_{0j} and β_{1j} are modeled as error in a linear model. As a consequence, the residual variance is larger and, therefore, the standard errors of point estimates, including means, coefficients of the model, and contrasts from the model, are larger. Here is the table of model coefficients of the data in Figure 19.1 fitted using a classical linear model

```
##               Estimate Std. Error  t value      Pr(>|t|)
## (Intercept) 10.0235749  0.2439544 41.08791 1.393185e-44
## TreatmentT+  0.6884787  0.3450036  1.99557 5.068466e-02
```

and a linear mixed model

```
##               Estimate Std. Error       df   t value      Pr(>|t|)
## (Intercept) 10.0235749  0.3954736 8.999737 25.345750 1.114804e-09
## TreatmentT+  0.6884787  0.2700797 9.000052  2.549169 3.123718e-02
```

The linear mixed model has increased precision (a smaller SE of the estimates) because it estimates the value of β_{0j} and β_{1j} for each block. The linear model does not estimate these parameters and the variance in these parameters is swept into the residual variance.

NHST blues – A **paired t-test** is equivalent to the special case of 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","T+"), 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
```

19.6 Linear mixed models are used to avoid pseudoreplication

19.7 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. Conequently, 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 paramters 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 19.2 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, $\mathrm{E}(b_j) \neq \beta_j$.
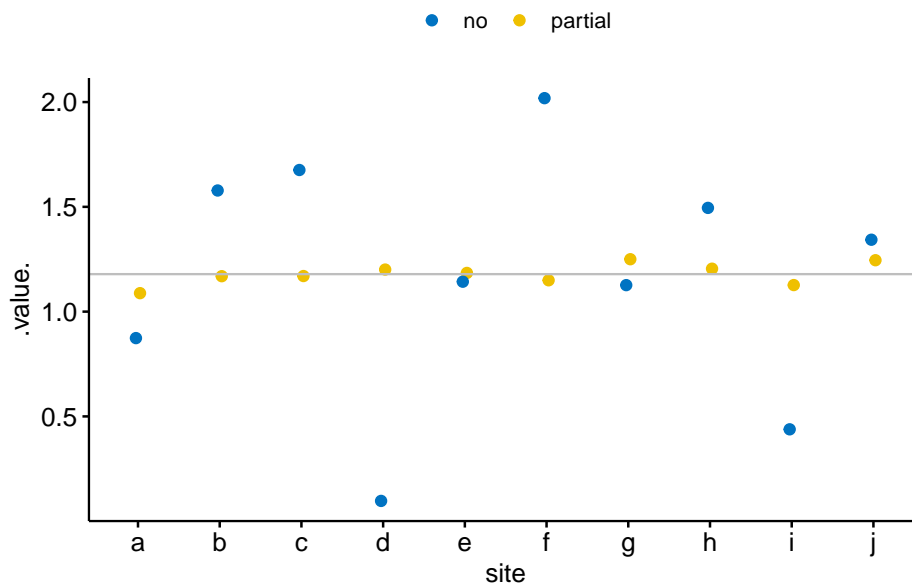
Figure 19.2: Shrinkage estimates of the treatment effect in a linear mixed model. The grey line is the estimate using complete pooling (so there is only one estimate which is assigned to each site). In general, the partial-pooling (linear mixed model) estimates (yellow) are generally closer to the complete 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.

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 19.2. 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 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 stasticians. 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 biostatics 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.

19.8   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.

19.8.1   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 vermetic
fn <- "VermetidSedimentData_ZillGilOsenberg_DRYAD.xlsx"
sheet_i <- "Coral Growth Rate Data"
file_path <- paste(data_path, folder, fn, sep="/")
coral <- data.table(read_excel(file_path, sheet=sheet_i))
setnames(coral, old=colnames(coral), new=clean_label(colnames(coral)))
coral[, Vermetids:=factor(Vermetids)]
coral[, Sediment:=factor(Sediment)]
```

lmer adds the random component to the formula. lme adds the random component as a separate argument

```
# to reproduce the results
# observation 2 should be excluded from the analysis
inc <- c(1, 3:nrow(coral))

# specification using lmer
# random intercept only
fit.lmer1 <- lmer(GrowthRate ~ Vermetids*Sediment + (1|Block), data=coral[inc])
# random intercept and slope
fit.lmer2 <- lmer(GrowthRate ~ Vermetids*Sediment + (Vermetids|Block) + (Sediment|Block), data=co
```

```
## boundary (singular) fit: see ?isSingular
```

```
# random intercept and slope
fit.lmer3 <- lmer(GrowthRate ~ Vermetids*Sediment + (Vermetids|Block) + (Sediment|Block), data=co
```

```
## boundary (singular) fit: see ?isSingular
```

```
# to include the interaction as a random effect we'd need subsampling within each factorial treat

# specification using lme
fit.lme <- lme(GrowthRate ~ Vermetids*Sediment, random= ~1|Block, data=coral[inc])

# results using lmer fit
coefficients(fit.lmer1)
```

```
## $Block
##    (Intercept)  Vermetids1 Sediment1 Vermetids1:Sediment1
## 1    1.205018 0.004655556 0.2852023           -0.7735467
## 2    1.336068 0.004655556 0.2852023           -0.7735467
## 3    1.213966 0.004655556 0.2852023           -0.7735467
## 4    1.262805 0.004655556 0.2852023           -0.7735467
## 6    1.320787 0.004655556 0.2852023           -0.7735467
## 7    1.201242 0.004655556 0.2852023           -0.7735467
## 8    1.314441 0.004655556 0.2852023           -0.7735467
## 9    1.199831 0.004655556 0.2852023           -0.7735467
```

```
## 10     1.361541 0.004655556 0.2852023              -0.7735467
##
## attr(,"class")
## [1] "coef.mer"
```

```
coefficients(summary(fit.lmer1))
```

```
##                        Estimate Std. Error       df     t value
## (Intercept)          1.268411111  0.1541680 30.42768  8.22745788
## Vermetids1           0.004655556  0.2091398 22.94243  0.02226049
## Sediment1            0.285202305  0.2160126 23.53130  1.32030428
## Vermetids1:Sediment1 -0.773546750  0.3006674 23.24638 -2.57276556
##                         Pr(>|t|)
## (Intercept)          3.129900e-09
## Vermetids1           9.824326e-01
## Sediment1            1.994327e-01
## Vermetids1:Sediment1 1.693404e-02
```

```
fit.emm <- emmeans(fit.lmer1, specs=c("Vermetids", "Sediment"))
summary(contrast(fit.emm, method="revpairwise", adjust="none"), infer=c(TRUE, TRUE))
```

```
##  contrast  estimate     SE   df lower.CL upper.CL t.ratio p.value
##  1,0 - 0,0  0.00466 0.209 23.0   -0.428   0.4373   0.022  0.9824
##  0,1 - 0,0  0.28520 0.217 23.6   -0.163   0.7330   1.316  0.2009
##  0,1 - 1,0  0.28055 0.217 23.6   -0.167   0.7283   1.294  0.2081
##  1,1 - 0,0 -0.48369 0.209 23.0   -0.916  -0.0511  -2.313  0.0300
##  1,1 - 1,0 -0.48834 0.209 23.0   -0.921  -0.0557  -2.335  0.0286
##  1,1 - 0,1 -0.76889 0.217 23.6   -1.217  -0.3211  -3.547  0.0017
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
```

```
fit.lmer1.ml <- lmer(GrowthRate ~ Vermetids*Sediment + (1|Block), data=coral[inc], REMI
# random intercept and slope
fit.lmer3.ml <- lmer(GrowthRate ~ Vermetids*Sediment + (Vermetids|Block) + (Sediment|B]
```

```
## boundary (singular) fit: see ?isSingular
```

```
AIC(fit.lmer1.ml)
```

```
## [1] 52.80349
```

```
AIC(fit.lmer3.ml)
```

```
## [1] 61.04197
```

The formula for `lmer`

Chapter 20

Linear models with heterogenous variance

20.1 gls

Appendix 1: Getting Started with R

20.2 Get your computer ready

20.2.1 Install R

R is the core software

Download R for your OS

20.2.2 Install R Studio

R Studio is a slick (very slick) GUI interface for developing R projects

Download R Studio Desktop

20.2.3 Resources for installing R and R Studio

On Windows

On a Mac

20.2.4 Install LaTeX

LaTeX ("la-tek") is necessary to use the pdf output of R Markdown.

On Windows

On a Mac

20.3   Start learning

20.3.1   Start with Data Camp Introduction to R

Data Camp: Introduction to R (free online course)

20.3.2   Then Move to Introduction to R Studio

R Studio Essentials, Programming Part 1 (Writing code in RStudio)

20.3.3   Develop your project with an R Studio Notebook

Getting Started with R Markdown

Introducing Notebooks with R Markdown

20.4   Getting Data into R

Getting your data into R

20.5   Additional R learning resources

Getting used to R, RStudio, and R Markdown

Link to list of R Studio webinars

Link to set of R package cheat sheets (amazing!)

Bookdown online books

20.6   Packages used extensively in this text

1. ggplot2
2. data.table
3. mvtnorm
4. lme4
5. nlme
6. emmeans
7. readxl
8. reshape2

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.

Statististical 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

Bibliography