Problems and Solutions in Applied Statistics (2nd ed)

Ken Butler

2023-06-08

Table of contents

Introduction **8**
Packages used somewhere in this book . . . 9

1 Getting used to R and R Studio **11**
1.1 Using R Studio online . . . 11
1.2 Using R Studio on your own computer . . . 17
1.3 Getting started . . . 17
1.4 Reading data from a file . . . 30
1.5 Reading files different ways . . . 34

2 Reading in data **44**
2.1 Orange juice . . . 44
2.2 Making soap . . . 44
2.3 Handling shipments . . . 45
2.4 Orange juice . . . 45
2.5 Making soap . . . 48
2.6 Handling shipments . . . 49

3 Data exploration **52**
3.1 North Carolina births . . . 52
3.2 More about the NC births . . . 52
3.3 Nenana, Alaska . . . 53
3.4 Computerized accounting . . . 53
3.5 Test scores in two classes . . . 54
3.6 Unprecedented rainfall . . . 55
3.7 Learning algebra . . . 55
3.8 North Carolina births . . . 56
3.9 More about the NC births . . . 68
3.10 Nenana, Alaska . . . 75
3.11 Computerized accounting . . . 83
3.12 Test scores in two classes . . . 92
3.13 Unprecedented rainfall . . . 105
3.14 Learning algebra . . . 114

4 One-sample inference **122**
4.1 Hunter-gatherers in Australia . . . 122

4.2 Buses to Boulder . . . 123
4.3 Length of gestation in North Carolina . . . 123
4.4 Inferring ice break-up in Nenana . . . 124
4.5 Diameters of trees . . . 124
4.6 One-sample cholesterol . . . 125
4.7 Hunter-gatherers in Australia . . . 126
4.8 Buses to Boulder . . . 131
4.9 Length of gestation in North Carolina . . . 138
4.10 Inferring ice break-up in Nenana . . . 142
4.11 Diameters of trees . . . 147
4.12 One-sample cholesterol . . . 155

5 Two-sample inference 167
5.1 Children and electronic devices . . . 167
5.2 Parking close to the curb . . . 167
5.3 Bell peppers and too much water . . . 168
5.4 Exercise and anxiety and bullying mice . . . 169
5.5 Diet and growth in boys . . . 169
5.6 Handspans of males and females . . . 170
5.7 The anchoring effect: Australia vs US . . . 170
5.8 Children and electronic devices . . . 171
5.9 Parking close to the curb . . . 179
5.10 Bell peppers and too much water . . . 193
5.11 Exercise and anxiety and bullying mice . . . 198
5.12 Diet and growth in boys . . . 204
5.13 Handspans of males and females . . . 209
5.14 The anchoring effect: Australia vs US . . . 216

6 Power and sample size 222
6.1 Simulating power . . . 222
6.2 Calculating power and sample size for estimating mean . . . 222
6.3 Simulating power for proportions . . . 223
6.4 Designing a study to have enough power . . . 224
6.5 Power and α in a skewed population . . . 224
6.6 Simulating power . . . 225
6.7 Calculating power and sample size for estimating mean . . . 233
6.8 Simulating power for proportions . . . 237
6.9 Designing a study to have enough power . . . 241
6.10 Power and α in a skewed population . . . 247

7 The sign test 258
7.1 Running a maze . . . 258
7.2 Chocolate chips . . . 259

7.3 The power of the sign test . . . 260
7.4 Ben Roethlisberger . . . 260
7.5 Six ounces of protein . . . 261
7.6 Running a maze . . . 262
7.7 Chocolate chips . . . 271
7.8 The power of the sign test . . . 276
7.9 Ben Roethlisberger . . . 288
7.10 Six ounces of protein . . . 301

8 Mood median test **309**
8.1 Sugar in breakfast cereals . . . 309
8.2 Fear of math . . . 310
8.3 Medical instructions . . . 310
8.4 Handspans revisited . . . 311
8.5 Sugar in breakfast cereals . . . 311
8.6 Fear of math . . . 316
8.7 Medical instructions . . . 324
8.8 Handspans revisited . . . 331

9 Matched pairs t and sign test **341**
9.1 Measuring body fat . . . 341
9.2 Throwing baseballs and softballs . . . 341
9.3 Throwing baseballs and softballs, again . . . 342
9.4 Changes in salary . . . 342
9.5 Body fat revisited . . . 343
9.6 The dentist and blood pressure . . . 343
9.7 French teachers . . . 344
9.8 Measuring body fat . . . 345
9.9 Throwing baseballs and softballs . . . 351
9.10 Throwing baseballs and softballs, again . . . 355
9.11 Changes in salary . . . 360
9.12 Body fat revisited . . . 366
9.13 The dentist and blood pressure . . . 370
9.14 French teachers . . . 376

10 Normal quantile plots **387**
10.1 Lengths of heliconia flowers . . . 387
10.2 Ferritin and normality . . . 388
10.3 Lengths of heliconia flowers . . . 388
10.4 Ferritin and normality . . . 396

11 Analysis of variance **400**
11.1 Movie ratings and lengths . . . 400

11.2 Deer and how much they eat . . . 400
11.3 Movie ratings again . . . 401
11.4 Atomic weight of carbon . . . 401
11.5 Can caffeine improve your performance on a test? . . . 402
11.6 Reggae music . . . 403
11.7 Watching TV and education . . . 404
11.8 Death of poets . . . 404
11.9 Religion and studying . . . 405
11.10 Movie ratings and lengths . . . 406
11.11 Deer and how much they eat . . . 411
11.12 Movie ratings again . . . 421
11.13 Atomic weight of carbon . . . 436
11.14 Can caffeine improve your performance on a test? . . . 446
11.15 Reggae music . . . 452
11.16 Watching TV and education . . . 463
11.17 Death of poets . . . 472
11.18 Religion and studying . . . 482

12 Writing reports **493**
12.1 Atomic weight of carbon . . . 493
12.2 Sparrowhawks . . . 494
12.3 Learning to code . . . 495
12.4 Treating dandruff . . . 495
12.5 Atomic weight of carbon . . . 497
12.6 Sparrowhawks . . . 506

13 Learning to code **517**
13.1 Introduction . . . 517
13.2 Data and pre-processing . . . 518
13.3 Analysis . . . 519
13.4 Conclusions (see note 9) . . . 521

14 Treating dandruff **528**
14.1 Introduction . . . 528
14.2 Exploratory analysis . . . 528
14.3 Analysis of Variance . . . 530
14.4 Assessment of Assumptions . . . 532
14.5 Conclusions . . . 534
14.6 End . . . 534

15 Tidying data **555**
15.1 Baseball and softball spaghetti . . . 555
15.2 Ethanol and sleep time in rats . . . 556

15.3 Growth of tomatoes . . . 556
15.4 Pain relief in migraine headaches (again) . . . 557
15.5 Location, species and disease in plants . . . 558
15.6 Mating songs in crickets . . . 558
15.7 Number 1 songs . . . 560
15.8 Bikes on College . . . 561
15.9 Feeling the heat . . . 562
15.10 Isoflavones . . . 563
15.11 Jocko's Garage . . . 564
15.12 Tidying electricity consumption . . . 565
15.13 Tidy blood pressure . . . 566
15.14 Baseball and softball spaghetti . . . 566
15.15 Ethanol and sleep time in rats . . . 574
15.16 Growth of tomatoes . . . 581
15.17 Pain relief in migraine headaches (again) . . . 589
15.18 Location, species and disease in plants . . . 597
15.19 Mating songs in crickets . . . 606
15.20 Number 1 songs . . . 620
15.21 Bikes on College . . . 631
15.22 Feeling the heat . . . 651
15.23 Isoflavones . . . 662
15.24 Jocko's Garage . . . 677
15.25 Tidying electricity consumption . . . 691
15.26 Tidy blood pressure . . . 704

16 Simple regression **718**
16.1 Rainfall in California . . . 718
16.2 Carbon monoxide in cigarettes . . . 719
16.3 Maximal oxygen uptake in young boys . . . 719
16.4 Facebook friends and grey matter . . . 721
16.5 Endogenous nitrogen excretion in carp . . . 721
16.6 Salaries of social workers . . . 722
16.7 Predicting volume of wood in pine trees . . . 723
16.8 Tortoise shells and eggs . . . 724
16.9 Roller coasters . . . 725
16.10 Running and blood sugar . . . 725
16.11 Calories and fat in pizza . . . 726
16.12 Where should the fire stations be? . . . 727
16.13 Making it stop . . . 727
16.14 Predicting height from foot length . . . 728
16.15 Rainfall in California . . . 729
16.16 Carbon monoxide in cigarettes . . . 741
16.17 Maximal oxygen uptake in young boys . . . 749

16.18 Facebook friends and grey matter . . . 756
16.19 Endogenous nitrogen excretion in carp . . . 763
16.20 Salaries of social workers . . . 774
16.21 Predicting volume of wood in pine trees . . . 788
16.22 Tortoise shells and eggs . . . 803
16.23 Roller coasters . . . 811
16.24 Running and blood sugar . . . 817
16.25 Calories and fat in pizza . . . 824
16.26 Where should the fire stations be? . . . 834
16.27 Making it stop . . . 843
16.28 Predicting height from foot length . . . 851

Introduction

This book contains a collection of problems, and my solutions to them, in applied statistics with R. These come from my courses STAC32, STAC33, and STAD29 at the University of Toronto Scarborough.

You will occasionally see question parts beginning with a *; this means that other question parts refer back to this one. (One of my favourite question strategies is to ask how two different approaches lead to the same answer, or more generally to demonstrate that there are different ways to see the same thing.)

Thanks to Dann Sioson for spotting some errors and making some useful suggestions.

If *you* see anything, file an issue on the Github page for now. Likely problems include:

- some LaTeX construction that I didn't catch (eg. block quotes)
- disappeared footnotes (that will show up as an apparently missing sentence in the text)
- references to "in class" or a lecture or a course by course number, which need to be eliminated (in favour of wording like "a previous course")
- references to other questions or question parts that are *wrong* (likely caused by *not* being "labels" or "refs" in the original LaTeX)
- my contorted English that is difficult to understand.

As I read through looking for problems like these, I realize that there ought to be a textbook that reflects my way of doing things. There isn't one (yet), though there are lecture notes. Current versions of these are at:

- the STAC32 website
- the STAC33 website
- the STAD29 website

A little background:

STAC32 is an introduction to R as applied to statistical methods that have (mostly) been learned in previous courses. This course is designed for students who have a second non-mathematical applied statistics course such as this. The idea is that students have already seen a little of regression and analysis of variance (and the things that precede them), and need mainly an introduction of how to run them in R.

STAC33 is an introduction to R, and applied statistics in general, for students who have a background in mathematical statistics. The way our courses are structured, these students have a strong mathematical background, but not very much experience in applications, which this course is designed to provide. The material covered is similar to STAC32, with a planned addition of some ideas in bootstrap and practical Bayesian statistics. There are some questions on these here.

STAD29 is an overview of a number of advanced statistical methods. I start from regression and proceed to some regression-like methods (logistic regression, survival analysis, log-linear frequency table analysis), then I go a little further with analysis of variance and proceed with MANOVA and repeated measures. I finish with a look at classical multivariate methods such as discriminant analysis, cluster analysis, principal components and factor analysis. I cover a number of methods in no great depth; my aim is to convey an understanding of what these methods are for, how to run them and how to interpret the results. Statistics majors and specialists cannot take this course for credit (they have separate courses covering this material with the proper mathematical background). D29 is intended for students in other disciplines who find themselves wanting to learn more statistics; we have an Applied Statistics Minor program for which C32 and D29 are two of the last courses.

Packages used somewhere in this book

The bottom lines are below used with the `conflicted` package: if a function by the name shown is in two or more packages, prefer the one from the package shown.

```
library(tidyverse)
library(smmr)
library(MASS)
library(nnet)
library(survival)
library(survminer)
library(car)
library(lme4)
library(ggbiplot)
library(ggrepel)
library(broom)
library(rpart)
library(bootstrap)
library(cmdstanr)
library(posterior)
library(bayesplot)
library(tmaptools)
```

```r
library(leaflet)
library(conflicted)
conflict_prefer("summarize", "dplyr")
conflict_prefer("select", "dplyr")
conflict_prefer("filter", "dplyr")
conflict_prefer("mutate", "dplyr")
conflict_prefer("count", "dplyr")
conflict_prefer("arrange", "dplyr")
conflict_prefer("rename", "dplyr")
conflict_prefer("id", "dplyr")
```

All of these packages are on CRAN, and may be installed via the usual `install.packages`, with the exceptions of:

- `smmr` on Github: install with

```r
devtools::install_github("nxskok/smmr")
```

- `ggbiplot` on Github: install with

```r
devtools::install_github("vqv/ggbiplot")
```

- `cmdstanr`, `posterior`, and `bayesplot`: install with

```r
install.packages("cmdstanr",
                 repos = c("https://mc-stan.org/r-packages/",
                           getOption("repos")))
install.packages("posterior",
                 repos = c("https://mc-stan.org/r-packages/",
                            getOption("repos")))
install.packages("bayesplot",
                 repos = c("https://mc-stan.org/r-packages/",
                            getOption("repos")))
```

1 Getting used to R and R Studio

Don't forget `library(tidyverse)` first if you need it (you probably will).

1.1 Using R Studio online

(a) Point your web browser at http://r.datatools.utoronto.ca. Click on the button to the left of "R Studio" (it will show blue), click the orange Log in to Start, and log in using your UTorID and password.

Solution

This is about what you should see first, before you click the orange thing:

You will see a progress bar as things start up, and then you should see something like this:

This is R Studio, ready to go.

If you are already logged in to something else on the same browser that uses your UTorID and password, you may come straight here without needing to log in again.

■

(b) Take a look around, and create a new Project. Give the new project any name you like.

Solution

Select File and New Project to get this:

Click on New Directory (highlighted blue on mine). This will create a new folder to put your new project in, which is usually what you want to do. The idea is that a project is a container for a larger collection of work, such as all your assignments in this course. That brings you to this:

where you click on New Project (highlighted on mine), and:

Give your project a name, as I did. Then click Create Project. At this point, R Studio will be restarted in your new project. You can tell which project you are in by looking top right, and you'll see the name of your project next to the R symbol:

■

(c) One last piece of testing: find the Console window (which is probably on the left). Click

next to the blue >, and type `library(tidyverse)`. Press Enter.

Solution

It may think a bit, and then you'll see something like this:

```
Console   Terminal ×   Jobs ×
~/

R version 4.0.5 (2021-03-31) -- "Shake and Throw"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> library(tidyverse)
── Attaching packages ─────────────────────────────── tidyverse 1.3.1 ──
✓ ggplot2 3.3.5     ✓ purrr   0.3.4
✓ tibble  3.1.6     ✓ dplyr   1.0.8
✓ tidyr   1.2.0     ✓ stringr 1.4.0
✓ readr   2.1.2     ✓ forcats 0.5.1
── Conflicts ────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
>
```

Aside: I used to use a cloud R Studio called `rstudio.cloud`. If you see or hear any references to that, it means the same thing as R Studio on `r.datatools` or `jupyter`. (You can still use

`rstudio.cloud` if you want; it used to be completely free, but now the free tier won't last you very long; the `utoronto.ca`link is free as long as you are at U of T.) I'm trying to get rid of references to R Studio Cloud as I see them, but I am bound to miss some, and in the lecture videos they are rather hard to find.

Now we can get down to some actual work.

■

1.2 Using R Studio on your own computer

This is not required now, but you may wish to do this now or later so that you are not fighting for resources on the `r.datatools` server at busy times (eg. when an assignment is due).

Follow the instructions here to install R Studio on your computer, then start R Studio (which itself starts R).

Once you have this working, you can use it for any of the following questions, in almost exactly the same way as the online R (I will explain any differences).

1.3 Getting started

This question is to get you started using R.

(a) Start R Studio on `r.datatools` (or on your computer), in some project. (If you started up a new project in the previous question and are still logged in, use that; if not, create a new project with File, New Project, and New Directory. Then select New Project and give it a name. Click Create Project. This will give you an empty workspace to start from.)

Solution

You ought to see something like this:

There should be one thing on the left half, and at the top right it'll say "Environment is empty".

Extra: if you want to tweak things, select Tools (at the top of the screen) and from it Global Options, then click Appearance. You can make the text bigger or smaller via Editor Font Size, and choose a different colour scheme by picking one of the Editor Themes (which previews on the right). My favourite is Tomorrow Night Blue. Click Apply or OK when you have found something you like. (I spend a lot of time in R Studio, and I like having a dark background to be easier on my eyes.)

■

(b) We're going to do some stuff in R here, just to get used to it. First, make a Quarto document by selecting File, New File and Quarto Document.

Solution

In the first box that pops up, you'll be invited to give your document a title. Make something up for now.

The first time, you might be invited to "install some packages" to make the document thing work.[1] Let it do that by clicking Yes. After that, you'll have this:

[1]Especially if you are on your own computer.

A couple of technical notes:

- this should be in the top left pane of your R Studio now, with the Console below it.
- At the top of the file, between the two lines with three hyphens (minus signs, whatever), is some information about the document, known in the jargon as a YAML block, any of which you can change:
  - the title is whatever title you gave your document
  - the `format`is what the *output* is going to be (in this case, HTML like a webpage, which is mostly what we'll be using)
  - there is a visual editor that looks like Notion or a bit like a Google doc (the default), and also a Source editor which gives you more control, and shows that underlying the document is a thing called R Markdown (which is a code for writing documents).
- My *document* is called "My awesome title", but the *file* in which the document lives is still untitled because I haven't saved it yet. See right at the top.

■

(c) You can delete the template code below the YAML block now (that is, everything from the title "Quarto" to the end). Somewhere in the space opened up below the YAML block (it might say "Heading 2", greyed out), type a /. This, like Notion, gives you a list of things to choose from to insert there. Pressing Enter will insert a "code chunk", sometimes known as a "code cell". We are going to use this in a moment.

Solution

Something like this:

```
---
title: "my awesome title"
format: html
editor: visual
---
```

```
{r}
```

The {r} at the top of the code chunk means that the code that will go in there will be R code (you can also have a Python code chunk, among others).

■

(d) On the line below the `{r}`, type these two lines of code into the chunk in the Quarto document:

```
library(tidyverse)
mtcars
```

Solution

What this will do: get hold of a built-in data set with information about some different models of car, and display it.

```
---
title: "my awesome title"
format: html
editor: visual
---
```

```
{r}
mtcars
```

In approximately five seconds, you'll be demonstrating that for yourself.

■

(e) Run this command. To do that, look at the top right of your code chunk block (shaded in a slightly different colour). You should see a down arrow and a green "play button". Click the play button. This will run the code, and show the output below the code chunk.

Solution

Here's what I get (yours should be the same):

```
{r}
mtcars
```

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>
Mazda...	21.0	6	160.0	110	3.90	2.620	16.46
Mazda...	21.0	6	160.0	110	3.90	2.875	17.02
Datsun...	22.8	4	108.0	93	3.85	2.320	18.61
Hornet...	21.4	6	258.0	110	3.08	3.215	19.44
Hornet...	18.7	8	360.0	175	3.15	3.440	17.02
Valiant	18.1	6	225.0	105	2.76	3.460	20.22
Duster...	14.3	8	360.0	245	3.21	3.570	15.84
Merc 2...	24.4	4	146.7	62	3.69	3.190	20.00
Merc 2...	22.8	4	140.8	95	3.92	3.150	22.90
Merc 2...	19.2	6	167.6	123	3.92	3.440	18.30

1-10 of 32 rows | 1-8 of 11 columns Previous 1 2 3 4 Next

This is a rectangular array of rows and columns, with individuals (here, cars) in rows and variables in columns, known as a "dataframe". When you display a dataframe in an Quarto document, you see 10 rows and as many columns as will fit on the screen. At the bottom, it says how many rows and columns there are altogether (here 32 rows and 11 columns), and which ones are being displayed.

You can see more rows by clicking on Next, and if there are more columns, you'll see a little arrow next to the rightmost column (as here next to `am`) that you can click on to see more columns. Try it and see. Or if you want to go to a particular collection of rows, click one of the numbers between Previous and Next: 1 is rows 1–10, 2 is rows 11–20, and so on.

The column on the left without a header (containing the names of the cars) is called "row names". These have a funny kind of status, kind of a column and kind of not a column; usually, if we need to use the names, we have to put them in a column first.

In future solutions, rather than showing you a screenshot, expect me to show you something like this:

```r
library(tidyverse)
mtcars
```

```
# A tibble: 32 x 12
   car           mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Mazda RX4    21       6  160    110  3.9   2.62  16.5     0     1     4     4
 2 Mazda RX4 ~  21       6  160    110  3.9   2.88  17.0     0     1     4     4
 3 Datsun 710   22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
 4 Hornet 4 D~  21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
 5 Hornet Spo~  18.7     8  360    175  3.15  3.44  17.0     0     0     3     2
 6 Valiant      18.1     6  225    105  2.76  3.46  20.2     1     0     3     1
 7 Duster 360   14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
 8 Merc 240D    24.4     4  147.    62  3.69  3.19  20       1     0     4     2
 9 Merc 230     22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
10 Merc 280     19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
# i 22 more rows
```

The top bit is the code, the bottom bit the output. In this kind of display, you only see the first ten rows (by default).[2]

If you don't see the "play button", make sure that what you have really is a code chunk. (I often accidentally delete one of the special characters above or below the code chunk). If you can't figure it out, delete this code chunk and make a new one. Sometimes R Studio gets confused.

On the code chunk, the other symbols are the settings for this chunk (you have the choice to display or not display the code or the output or to not actually run the code). The second one, the down arrow, runs all the chunks prior to this one (but not this one).

Your output has its own little buttons (as seen on the screenshot). The first one pops the output out into its own window; the second one shows or hides the output, and the third one deletes the output (so that you have to run the chunk again to get it back). Experiment. You can't do much damage here.

■

(f) Something a little more interesting: `summary` obtains a summary of whatever you feed it (the five-number summary plus the mean for numerical variables). Obtain this for our data frame. To do this, create a new code chunk below the previous one, type `summary(mtcars)` into the code chunk, and run it.

[2]This document was actually produced by literally running this code, a process known as "rendering", which we will learn about shortly.

Solution

This is what you should see:

or the other way:

```r
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat              wt              qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am               gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
```

```
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

For the gas mileage column `mpg`, the mean is bigger than the median, and the largest value is unusually large compared with the others, suggesting a distribution that is skewed to the right.

There are 11 numeric (quantitative) variables, so we get the five-number summary plus mean for each one. Categorical variables, if we had any here, would be displayed a different way.

■

(g) Let's make a histogram of the gas mileage data. Type the code below into another new code chunk, and run it:

```
ggplot(mtcars, aes(x = mpg)) + geom_histogram(bins = 8)
```

The code looks a bit wordy, but we'll see what all those pieces do later in the course (like, maybe tomorrow).

Solution

This is what you should see:

```
ggplot(mtcars, aes(x = mpg)) + geom_histogram(bins = 8)
```

The long right tail supports our guess from before that the distribution is right-skewed.

■

(h) Some aesthetics: Add some narrative text above and below your code chunks. Above the code chunk is where you say what you are going to do (and maybe why you are doing it), and below is where you say what you conclude from the output you just obtained. I find it looks better if you have a blank line above and below each code chunk.

Solution

This is what I wrote (screenshot), with none of the code run yet. My `library(tidyverse)` line seems to have disappeared, but yours should still be there:

```
---
title: "my awesome title"
format: html
editor: visual
---
```

Here is the built-in data set `mtcars`, with some information about 32 different cars:

```{r}
mtcars
```

We look at a numerical summary of each of the eleven (quantitative) variables:

```{r}
summary(mtcars)
```

For the `mpg` column, the mean is larger than the median, so we suspect that the distribution will be right-skewed. This can be confirmed by looking at a histogram:

```{r}
ggplot(mtcars, aes(x = mpg)) + geom_histogram(bins = 8)
```

The longer upper tail indicates a right skew.

■

(i) Save your Quarto document (the usual way with File and Save). This saves it *on the jupyter servers* (and not on your computer). This means that when you come back to it later, even from another device, this document will still be available to you. (If you are running R Studio on your own computer, it is much simpler: the Quarto document is on that computer, in the folder associated with the project you created.)

Now click Render. This produces a pretty HTML version of your Quarto document. This will appear in a new tab of your web browser, which you might need to encourage to appear (if you have a pop-up blocker) by clicking a Try Again.

Solution

If there are any errors in the rendering process, these will appear in the Render tab. The error message will tell you where in your document your error was. Find it and correct it.[3] Otherwise, you should see your document.

Extra 1: the rendering process as you did it doesn't produce that nice display of a dataframe that I had in one of my screenshots. To get that, alter the YAML block to read:

```
format:
  html:
     df-print: paged
```

This way, anyone reading your document can actually page through the dataframes you display in the same way that you did, to check that they contain the right things.

Extra 2: you might prefer to have a preview of your document within R Studio. To make this happen, look for the gear wheel to the right of Render. Click the arrow beside it, and in the drop-down, click on Preview in Viewer Pane. Render again, and you'll see the rendered version of your document in a Viewer pane on the right. This puts the thing you're writing and what it will look like side by side.

Extra 3: you might be annoyed at having to remember to save things. If you are, you can enable auto-saving. To do this, go to Tools and select Global Options. Select Code (on the left) and Saving (at the top). Click on Automatically Save when editor loses focus, to put a check mark in the box on the left of it. Change the pull-down below that to Save and Write Changes. Click OK. Now, as soon as you pause for a couple of seconds, everything unsaved will be saved.

■

(j) Practice handing in your rendered Quarto document, as if it were an assignment that was worth something. (It is good to get the practice in a low-stakes situation, so that you'll know what to do next week.)

Solution

There are two steps: download the HTML file onto your computer, and then handing it in on Quercus. To download: find the HTML file that you want to download in the Files pane on

[3] A big part of coding is dealing with errors. You will forget things, and it is fine. In the same way that it doesn't matter how many times you get knocked down, it's key that you get up again each time: it doesn't matter how many errors you made, it's key that you fix them. If you want something to sing along with while you do this, I recommend this.

the right. You might need to click on Files at the top, especially if you had a Viewer open there before:

I called my Quarto document `awesome`and the file I was working on was called `awesome.qmd`(which stands for "Quarto Markdown"). That's the file I had to render to produce the output. My output file itself is called `awesome.html`.That's the file I want to hand in. If you called your file something different when you saved it, that's the thing to look for: there should be something ending in `.qmd` and something with the same first part ending in `.html`.

Click the checkbox to the left of the HTML file. Now click on More above the bottom-right pane. This pops up a menu from which you choose Export. This will pop up another window called Export Files, where you put the name that the file will have on your computer. (I usually leave the name the same.) Click Download. The file will go to your Downloads folder, or wherever things you download off the web go.

Now, to hand it in. Open up Quercus at `q.utoronto.ca`, log in and navigate to this course. Click Assignments. Click (the title of) Assignment 0. There is a big blue Start Assignment button top right. Click it. You'll get a File Upload at the bottom of the screen. Click Choose File and find the HTML file that you downloaded. Click Open (or equivalent on your system). The name of the file should appear next to Choose File. Click Submit Assignment. You'll see Submitted at the top right, and below that is a Submission Details window and the file you uploaded.

You should be in the habit of *always* checking what you hand in, by downloading it again and looking at it to make sure it's what you thought you had handed in.

If you want to try this again, you can try again as many times as you like, by making a New Attempt. (For the real thing, you can use this if you realize you made a mistake in something you submitted. The graders' instructions, for the real thing, are to grade the *last* file submitted, so in that case you need to make sure that the last thing submitted before the due date includes *everything* that you want graded. Here, though, it doesn't matter.)

■

(k) Optional extra. Something more ambitious: make a scatterplot of gas mileage `mpg`, on the y axis, against horsepower, `hp`, on the x-axis.

Solution

That goes like this. I'll explain the steps below.

```
library(tidyverse)
ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point()
```

This shows a somewhat downward trend, which is what you'd expect, since a larger `hp` value means a more powerful engine, which will probably consume more gas and get *fewer* miles per gallon. As for the code: to make a `ggplot` plot, as we will shortly see in class, you first need a `ggplot` statement that says what to plot. The first thing in a `ggplot` is a data frame (`mtcars` here), and then the `aes` says that the plot will have `hp` on the x-axis and `mpg` on the y-axis, taken from the data frame that you specified. That's all of the what-to-plot. The last thing is how to plot it; `geom_point()` says to plot the data values as points.

You might like to add a regression line to the plot. That is a matter of adding this to the end of the plotting command:

```
ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point() + geom_smooth(method="lm")
```

```
`geom_smooth()` using formula = 'y ~ x'
```

The line definitely goes downhill. Decide for yourself how well you think a line fits these data.

■

1.4 Reading data from a file

In this question, we read a file from the web and do some descriptive statistics and a graph. This is very like what you will be doing on future assignments, so it's good to practice it now.

Take a look at the data file at http://ritsokiguess.site/datafiles/jumping.txt. These are measurements on 30 rats that were randomly made to do different amounts of jumping by group (we'll see the details later in the course). The control group did no jumping, and the other groups did "low jumping" and "high jumping". The first column says which jumping group each rat was in, and the second is the rat's bone density (the experimenters' supposition was that more jumping should go with higher bone density).

(a) What are the two columns of data separated by? (The fancy word is "delimited").

Solution

Exactly one space. This is true all the way down, as you can check.

■

(b) Make a new Quarto document. Leave the YAML block, but get rid of the rest of the template document. Start with a code chunk containing `library(tidyverse)`. Run it.

Solution

You will get either the same message as before or nothing. (I got nothing because I had already loaded the `tidyverse` in this session.)

■

(c) Put the URL of the data file in a variable called `my_url`. Then use `read_delim` to read in the file. (See solutions for how.) `read_delim` reads data files where the data values are always separated by the same single character, here a space. Save the data frame in a variable `rats`.

Solution

Like this:

```
my_url <- "http://ritsokiguess.site/datafiles/jumping.txt"
rats <- read_delim(my_url," ")
```

```
Rows: 30 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): group
dbl (1): density

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The second thing in `read_delim` is the thing that separates the data values. Often when you use `read_delim` it'll be a space.

Hint: to get the file name into `my_url`, the best way is to *right*-click on the link, and select Copy Link Address (or equivalent in your browser). That puts in on your clipboard. Then make a code chunk and put this in it (you'll probably only need to type one quote symbol, because R Studio will supply the other one):

```
my_url <- ""
```

then put the cursor between the two quote symbols and paste. This is better than selecting the URL in my text and then copy-pasting that because odd things happen if it happens to span two lines on your screen. (URLs tend to be rather long, so this is not impossible.)

■

(d) Take a look at your data frame, by making a new code chunk and putting the data frame's name in it (as we did with `mtcars`).

Solution

```
rats
```

```
# A tibble: 30 x 2
   group   density
   <chr>     <dbl>
 1 Control     611
 2 Control     621
 3 Control     614
 4 Control     593
 5 Control     593
 6 Control     653
 7 Control     600
 8 Control     554
 9 Control     603
10 Control     569
# i 20 more rows
```

There are 30 rows and two columns, as there should be.

■

(e) Find the mean bone density for rats that did each amount of jumping.

Solution

This is something you'll see a lot: `group_by` followed by `summarize`. Reminder: to get that funny thing with the percent signs (called the "pipe symbol"), type control-shift-M (or equivalent on a Mac):

```
rats %>% group_by(group) %>%
summarize(m = mean(density))
```

```
# A tibble: 3 x 2
  group        m
  <chr>    <dbl>
1 Control   601.
2 Highjump  639.
3 Lowjump   612.
```

The mean bone density is clearly highest for the high jumping group, and not much different between the low-jumping and control groups.

■

(f) Make a boxplot of bone density for each jumping group.

Solution

On a boxplot, the groups go across and the values go up and down, so the right syntax is this:

```
ggplot(rats, aes(x=group, y=density)) + geom_boxplot()
```

Given the amount of variability, the control and low-jump groups are very similar (with the control group having a couple of outliers), but the high-jump group seems to have a consistently higher bone density than the others.

This is more or less in line with what the experimenters were guessing, but it seems that it has to be high jumping to make a difference.

You might recognize that this is the kind of data where we would use analysis of variance, which we will do later on in the course: we are comparing several (here three) groups.

■

1.5 Reading files different ways

This question is about different ways of reading data files. If you're working online (using `r.datatools` or similar), start at the beginning. If you're using R Studio running on your own computer, start at part (here).

(a) Log in to `r.datatools.utoronto.ca`. Open up a project (or start a new one), and watch the spinning circles for a few minutes. When that's done, create a new Quarto Document with File, New File, Quarto Document. Delete the "template" document, but not the top lines with `title:` and `output:` in them. Add a code chunk that contains `library(tidyverse)` and run it.

Solution

So far you (with luck) have something that looks like this:

If you have an error rather than that output, you probably need to install the `tidyverse` first. Make another code chunk, containing

```
install.packages("tidyverse")
```

and run it. Wait for it to finish. It may take a while. If it completes successfully (you might see the word DONE at the end), delete that code chunk (you don't need it any more) and try again with the `library(tidyverse)` chunk. It should run properly this time.

■

(b) * The easiest kind of files to read in are ones on the Internet, with a URL address that begins with `http` or `https`. I have a small file at link. Click the link to see it, and keep the tab open for the next part of this question. This is a text file with three things on each line, each separated by exactly one space. Read the data file into a data frame, and display your data frame.

Solution

Data values separated by exactly one space is the kind of thing that `read_delim` reads, so make another code chunk and fill it with this:

```
my_url <- "http://ritsokiguess.site/datafiles/testing.txt"
testing <- read_delim(my_url, " ")
```

```
Rows: 6 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): g
dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
testing
```

```
# A tibble: 6 x 3
      x     y g
  <dbl> <dbl> <chr>
1     1    10 a
2     2    11 b
3     3    14 a
4     4    13 b
5     5    18 a
6     6    21 b
```

When you run that, you'll see something like my output. The first part is `read_delim` telling you what it saw in the file: two columns of (whole) numbers and one column of text. The top line of the file is assumed to contain names, which are used as the names of the columns of your data frame. The bottom part of the output, obtained by putting the name of the data frame on a line by itself in your code chunk, is what the data frame actually looks like. You

ought to get into the habit of eyeballing it and checking that it looks like the values in the data file.

The things on the left side of the `<-` symbol (that is meant to look like an arrow pointing left) are variables that you are creating in R. You get to choose the names for them. My habit is to use `my_url` for URLs of files that I am going to read in, and (usually) to give my data frames names that say something about what they contain, but this is your choice to make.

■

(c) You might have data in a file on your own computer. To read data from such a file, you first have to *upload* it to `r.datatools`, and then read it from there. To practice this: open a text editor (like Notepad or TextEdit). Go back to the web browser tab containing the data you used in the previous part. Copy the data from there and paste it into the text editor. Save it somewhere on your computer (like the Desktop). Upload that file, read in the data and verify that you get the right thing. (For this last part, see the Solution.)

Solution

I copied and pasted the data, and saved it in a file called `testing.txt` on my computer. I'm assuming that you've given it a similar name. Then go back to `r.datatools`. You should have a Files pane bottom right. If you don't see a pane bottom right at all, press Control-Shift-0 to show all the panes. If you see something bottom right but it's not Files (for example a plot), click the Files tab, and you should see a list of things that look like files, like this:

Click the Upload button (next to New Folder), click Choose File. Use the file finder to track down the file you saved on your computer, then click OK. The file should be uploaded to the same folder on `r.datatools` that your project is, and appear in the Files pane bottom right.

To read it in, you supply the file name to `read_delim` thus:

```
testing2 <- read_delim("testing.txt", " ")
```

```
Rows: 6 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): g
dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

and then you look at it in the same way as before:

```
testing2
```

```
# A tibble: 6 x 3
      x     y g
  <dbl> <dbl> <chr>
1     1    10 a
2     2    11 b
3     3    14 a
4     4    13 b
5     5    18 a
6     6    21 b
```

Check.

■

(d) You might have a spreadsheet on your computer. Create a `.csv` file from it, and use the ideas of the last part to read it into R Studio.

Solution

Open the spreadsheet containing the data you want to read into R. If there are several sheets in the workbook, make sure you're looking at the right one. Select File, Save As, select "CSV" or "comma-separated values" and give it a name. Save the resulting file somewhere.

Then follow the same steps as the previous part to upload it to your project on R Studio Cloud. (If you look at the actual file, it will be plain text with the data values having commas between them, as the name suggests. You can open the file in R Studio by clicking on it in the Files pane; it should open top left.)

The final step is to read it into an R data frame. This uses `read_csv`; there are several `read_` functions that read in different types of file, and you need to use an appropriate one.

My spreadsheet got saved as `cars.csv`, so:

```
cars <- read_csv("cars.csv")
```

```
Rows: 38 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): Car, Country
dbl (4): MPG, Weight, Cylinders, Horsepower

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
cars
```

```
# A tibble: 38 x 6
   Car                  MPG Weight Cylinders Horsepower Country
   <chr>              <dbl>  <dbl>     <dbl>      <dbl> <chr>
 1 Buick Skylark       28.4   2.67         4         90 U.S.
 2 Dodge Omni          30.9   2.23         4         75 U.S.
 3 Mercury Zephyr      20.8   3.07         6         85 U.S.
 4 Fiat Strada         37.3   2.13         4         69 Italy
 5 Peugeot 694 SL      16.2   3.41         6        133 France
 6 VW Rabbit           31.9   1.92         4         71 Germany
 7 Plymouth Horizon    34.2   2.2          4         70 U.S.
 8 Mazda GLC           34.1   1.98         4         65 Japan
 9 Buick Estate Wagon  16.9   4.36         8        155 U.S.
10 Audi 5000           20.3   2.83         5        103 Germany
# i 28 more rows
```

Some information about different types of cars.

You are now done with this question.

■

(e) * Start here if you downloaded R and R Studio and they are running on your own computer. Open a web browser and point it at link. Click the link to see it, and keep the tab open for the next part of this question. This is a text file with three things on each line, each separated by exactly one space. Read the data file into a data frame, and display your data frame.

Solution

Data values separated by exactly one space is the kind of thing that `read_delim` reads, so make another code chunk and fill it with this:

```
my_url <- "http://ritsokiguess.site/datafiles/testing.txt"
testing <- read_delim(my_url, " ")
```

```
Rows: 6 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): g
dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
testing
```

```
# A tibble: 6 x 3
      x     y g
  <dbl> <dbl> <chr>
1     1    10 a
2     2    11 b
3     3    14 a
4     4    13 b
5     5    18 a
6     6    21 b
```

When you run that, you'll see something like my output. The first part is `read_delim` telling you what it saw in the file: two columns of (whole) numbers and one column of text. The top line of the file is assumed to contain names, which are used as the names of the columns of your data frame. The bottom part of the output, obtained by putting the name of the data frame on a line by itself in your code chunk, is what the data frame actually looks like. You

ought to get into the habit of eyeballing it and checking that it looks like the values in the data file.

The things on the left side of the `<-` symbol (that is meant to look like an arrow pointing left) are variables that you are creating in R. You get to choose the names for them. My habit is to use `my_url` for URLs of files that I am going to read in, and (usually) to give my data frames names that say something about what they contain, but this is your choice to make.

(This part is exactly the same whether you are running R Studio on `r.datatools` or have R Studio running on your computer. A remote file is obtained in exactly the same way regardless.)

■

(f) You might have data in a file on your own computer. To read data from such a file, R has to know where to find it. Each R project lives in a folder, and one way of specifying where a data file is is to give its complete path relative to the folder that R Studio is running its current project in. This is rather complicated, so we will try a simpler way. To set this up, open a text editor (like Notepad or TextEdit). Go back to the web browser tab containing the data you used in the previous part. Copy the data from there and paste it into the text editor. Save it somewhere on your computer (like the Desktop). Follow the steps in the solution below to read the data into R.

Solution

I copied and pasted the data, and saved it in a file called `testing.txt` on my computer. I'm assuming that you've given it a similar name. Go back to R Studio. Create a new code chunk containing this:

```
f <- file.choose()
```

Run this code chunk. You'll see a file chooser. Find the file you saved on your computer, and click Open (or OK or whatever you see). This saves what R needs to access the file in the variable `f`. If you want to, you can look at it:

```
f
```

and you'll see what looks like a file path in the appropriate format for your system (Windows, Mac, Linux). To read the data in, you supply the file path to `read_delim` thus:

```
testing2 <- read_delim(f, " ")
```

and then you look at it in the same way as before:

```
testing2
```

```
# A tibble: 6 x 3
      x     y g
  <dbl> <dbl> <chr>
1     1    10 a
2     2    11 b
3     3    14 a
4     4    13 b
5     5    18 a
6     6    21 b
```

Check.

■

(g) You might have a spreadsheet on your computer. Create a `.csv` file from it, and use the ideas of the last part to read it into R Studio.

Solution

Open the spreadsheet containing the data you want to read into R. If there are several sheets in the workbook, make sure you're looking at the right one. Select File, Save As, select "CSV" or "comma-separated values" and give it a name. Save the resulting file somewhere.

Then read it into an R data frame. This uses `read_csv`; there are several `read_` functions that read in different types of file, and you need to use an appropriate one. Before that, though, again run

```
f <- file.choose()
```

to find the `.csv` file on your computer, and then

```
cars <- read_csv(f)
```

to read it in. My spreadsheet was

```
cars
```

```
# A tibble: 38 x 6
   Car                  MPG Weight Cylinders Horsepower Country
   <chr>              <dbl>  <dbl>     <dbl>      <dbl> <chr>
 1 Buick Skylark       28.4   2.67         4         90 U.S.
 2 Dodge Omni          30.9   2.23         4         75 U.S.
 3 Mercury Zephyr      20.8   3.07         6         85 U.S.
 4 Fiat Strada         37.3   2.13         4         69 Italy
```

```
 5 Peugeot 694 SL      16.2   3.41         6        133 France
 6 VW Rabbit           31.9   1.92         4         71 Germany
 7 Plymouth Horizon    34.2   2.2          4         70 U.S.
 8 Mazda GLC           34.1   1.98         4         65 Japan
 9 Buick Estate Wagon  16.9   4.36         8        155 U.S.
10 Audi 5000           20.3   2.83         5        103 Germany
# i 28 more rows
```

Some information about different types of cars.

■

2 Reading in data

In this chapter, and from here on, the questions are first, and then my answers are in the second appearances of the problems with the same name. It is much better for your learning to spend some time thinking about how you would solve these problems, and after *that* you can see how I did it.

```
library(tidyverse)
```

2.1 Orange juice

The quality of orange juice produced by a manufacturer (identity unknown) is constantly being monitored. The manufacturer has developed a "sweetness index" for its orange juice, for which a higher value means sweeter juice. Is the sweetness index related to a chemical measure such as the amount of water-soluble pectin (parts per million) in the orange juice? Data were obtained from 24 production runs, and the sweetness and pectin content were measured for each run. The data are in link. Open that link up now. You can click on that link just above to open the file.

(a) The data values are separated by a space. Use the appropriate Tidyverse function to read the data directly from the course website into a "tibble".

(b) Take a look at what got read in. Do you have data for all 24 runs?

(c) In your data frame, where did the column (variable) names come from? How did R know where to get them from?

2.2 Making soap

A company operates two production lines in a factory for making soap bars. The production lines are labelled A and B. A production line that moves faster may produce more soap, but may possibly also produce more "scrap" (that is, bits of soap that can no longer be made into soap bars and will have to be thrown away).

The data are in link.

(a) Read the data into R. Display the data.

(b) There should be 27 rows. Are there? What columns are there?

2.3 Handling shipments

A company called Global Electronics from time to time imports shipments of a certain large part used as a component in several of its products. The size of the shipment varies each time. Each shipment is sent to one of two warehouses (labelled A and B) for handling. The data in link show the `size` of each shipment (in thousands of parts) and the direct `cost` of handling it, in thousands of dollars. Also shown is the `warehouse` (A or B) that handled each shipment.

(a) Read the data into R and display your data frame.

(b) Describe how many rows and columns your data frame has, and what they contain.

My solutions follow:

2.4 Orange juice

The quality of orange juice produced by a manufacturer (identity unknown) is constantly being monitored. The manufacturer has developed a "sweetness index" for its orange juice, for which a higher value means sweeter juice. Is the sweetness index related to a chemical measure such as the amount of water-soluble pectin (parts per million) in the orange juice? Data were obtained from 24 production runs, and the sweetness and pectin content were measured for each run. The data are in link. Open that link up now. You can click on that link just above to open the file.

(a) The data values are separated by a space. Use the appropriate Tidyverse function to read the data directly from the course website into a "tibble".

Solution

Start with this (almost always):

```
library(tidyverse)
```

The appropriate function, the data values being separated by a space, will be `read_delim`. Put the URL as the first thing in `read_delim`, or (better) define it into a variable first:[1]

[1] I say "better" because otherwise the read line gets rather long. This way you read it as "the URL is some long thing that I don't care about especially, and I what I need to do is to read the data from that URL, separated by spaces."

```r
url <- "http://ritsokiguess.site/datafiles/ojuice.txt"
juice <- read_delim(url, " ")
```

```
Rows: 24 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (3): run, sweetness, pectin

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

`read_delim` (or `read_csv` or any of the others) tell you what variables were read in, and also tell you about any "parsing errors" where it couldn't work out what was what. Here, we have three variables, which is entirely consistent with the three columns of data values in the file.

`read_delim` can handle data values separated by *any* character, not just spaces, but the separating character, known as a "delimiter", does *not* have a default, so you have to say what it is, every time.

■

(b) Take a look at what got read in. Do you have data for all 24 runs?

Solution

Type the name of the data frame in a code chunk (a new one, or add it to the end of the previous one). Because this is actually a "tibble", which is what `read_delim` reads in, you'll only actually see the first 10 lines, but it will tell you how many lines there are altogether, and you can click on the appropriate thing to see the rest of it.

```r
juice
```

```
# A tibble: 24 x 3
     run sweetness pectin
   <dbl>     <dbl>  <dbl>
 1     1       5.2    220
 2     2       5.5    227
 3     3       6      259
 4     4       5.9    210
 5     5       5.8    224
 6     6       6      215
 7     7       5.8    231
```

```
 8     8       5.6    268
 9     9       5.6    239
10    10       5.9    212
# i 14 more rows
```

I appear to have all the data. If you want further convincing, click Next a couple of times to be sure that the runs go down to number 24.

■

(c) In your data frame, where did the column (variable) names come from? How did R know where to get them from?

Solution

They came from the top line of the data file, so we didn't have to specify them. This is the default behaviour of all the `read_` functions, so we don't have to ask for it specially.

Extra: in fact, if the top line of your data file is *not* variable names, *that's* when you have to say something special. The `read_` functions have an option `col_names` which can either be `TRUE` (the default), which means "read them in from the top line", `FALSE` ("they are not there, so make some up") or a list of column names to use. You might use the last alternative when the column names that are in the file are *not* the ones you want to use; in that case, you would also say `skip=1` to skip the first line. For example, with file `a.txt` thus:

```
a b
1 2
3 4
5 6
```

you could read the same data but call the columns `x` and `y` thus:

```r
read_delim("a.txt", " ", col_names = c("x", "y"), skip = 1)
```

```
Rows: 3 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# A tibble: 3 x 2
      x     y
  <dbl> <dbl>
1     1     2
2     3     4
3     5     6
```

■

2.5 Making soap

A company operates two production lines in a factory for making soap bars. The production lines are labelled A and B. A production line that moves faster may produce more soap, but may possibly also produce more "scrap" (that is, bits of soap that can no longer be made into soap bars and will have to be thrown away).

The data are in link.

(a) Read the data into R. Display the data.

Solution

Read directly from the URL, most easily:

```r
url <- "http://ritsokiguess.site/datafiles/soap.txt"
soap <- read_delim(url, " ")
```

```
Rows: 27 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): line
dbl (3): case, scrap, speed

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
soap
```

```
# A tibble: 27 x 4
    case scrap speed line
   <dbl> <dbl> <dbl> <chr>
 1     1   218   100 a
 2     2   248   125 a
 3     3   360   220 a
 4     4   351   205 a
 5     5   470   300 a
 6     6   394   255 a
 7     7   332   225 a
 8     8   321   175 a
 9     9   410   270 a
10    10   260   170 a
# i 17 more rows
```

■

(b) There should be 27 rows. Are there? What columns are there?

Solution

There are indeed 27 rows, one per observation. The column called `case` identifies each particular run of a production line (scroll down to see that it gets to 27 as well). Though it is a number, it is an identifier variable and so should not be treated quantitatively. The other columns (variables) are `scrap` and `speed` (quantitative) and `line` (categorical). These indicate which production line was used for each run, the speed it was run at, and the amount of scrap produced.

This seems like an odd place to end this question, but later we'll be using these data to draw some graphs.

■

2.6 Handling shipments

A company called Global Electronics from time to time imports shipments of a certain large part used as a component in several of its products. The size of the shipment varies each time. Each shipment is sent to one of two warehouses (labelled A and B) for handling. The data in link show the `size` of each shipment (in thousands of parts) and the direct `cost` of handling it, in thousands of dollars. Also shown is the `warehouse` (A or B) that handled each shipment.

(a) Read the data into R and display your data frame.

Solution

If you open the data file in your web browser, it will probably open as a spreadsheet, which is not really very helpful, since then it is not clear what to do with it. You could, I suppose, save it and upload it to R Studio Cloud, but it requires much less brainpower to open it directly from the URL:

```
url <- "http://ritsokiguess.site/datafiles/global.csv"
shipments <- read_csv(url)
```

```
Rows: 10 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): warehouse
dbl (2): size, cost

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

If you display your data frame and it looks like this, you are good (you can give the data frame any name):

```
shipments
```

```
# A tibble: 10 x 3
   warehouse  size  cost
   <chr>     <dbl> <dbl>
 1 A           225 12.0
 2 B           350 14.1
 3 A           150  8.93
 4 A           200 11.0
 5 A           175 10.0
 6 A           180 10.1
 7 B           325 13.8
 8 B           290 13.3
 9 B           400 15
10 A           125  7.97
```

■

(b) Describe how many rows and columns your data frame has, and what they contain.

Solution

It has 10 rows and 3 columns. *You need to say this.*

That is, there were 10 shipments recorded, and for each of them, 3 variables were noted: the size and cost of the shipment, and the warehouse it was handled at.

We will also be making some graphs of these data later.

■

3 Data exploration

```
library(tidyverse)
```

3.1 North Carolina births

The data in file link are about 500 randomly chosen births of babies in North Carolina. There is a lot of information: not just the weight at birth of the baby, but whether the baby was born prematurely, the ages of the parents, whether the parents are married, how long (in weeks) the pregnancy lasted (this is called the "gestation") and so on.

(a) Read in the data from the file into R, bearing in mind what type of file it is.

(b) From your output, verify that you have the right number of observations and that you have several variables. Which of your variables correspond to birthweight, prematureness and length of pregnancy? (You might have to make guesses based on the names of the variables.)

(c) The theory behind the t-test (which we do later) says that the distribution of birth weights should be (approximately) normally distributed. Obtain a histogram of the birth weights. Does it look approximately normal? Comment briefly. (You'll have to pick a number of bins for your histogram first. I don't mind very much what you pick, as long as it's not obviously too many or too few bins.)

3.2 More about the NC births

This is an exploration of some extra issues around the North Carolina births data set.

(a) How short does a pregnancy have to be, for the birth to be classified as "premature"? Deduce this from the data, by drawing a suitable graph or otherwise.

(b) Explore the relationship between birth weight and length of pregancy ("gestation") using a suitable graph. What do you see?

(c) Do a web search to find the standard (North American) definition of a premature birth. Does that correspond to what you saw in the data? Cite the website you used, for example by saying "according to `URL`, ...", with `URL` replaced by the address of the website you found.

3.3 Nenana, Alaska

Nenana, Alaska, is about 50 miles west of Fairbanks. Every spring, there is a contest in Nenana. A wooden tripod is placed on the frozen river, and people try to guess the exact minute when the ice melts enough for the tripod to fall through the ice. The contest started in 1917 as an amusement for railway workers, and has taken place every year since. Now, hundreds of thousands of people enter their guesses on the Internet and the prize for the winner can be as much as $300,000.

Because so much money is at stake, and because the exact same tripod is placed at the exact same spot on the ice every year, the data are consistent and accurate. The data are in link.

(a) Read the data into R. Note that the values are separated by *tabs* rather than spaces, so you'll need an appropriate `read_` to read it in.

(b) Find a way of displaying how many rows and columns your data frame has, and some of the values. Describe the first and last of the variables that you appear to have.

(c) Dates and times are awkward to handle with software. (We see more ways later in the course.) The column `JulianDate` expresses the time that the tripod fell through the ice as a fractional number of days since December 31. This enables the time (as a fraction of the way through the day) to be recorded as well, the whole thing being an ordinary number. Make a histogram of the Julian dates. Comment briefly on its shape.

(d) Plot `JulianDate` against `Year` on a scatterplot. What recent trends, if any, do you see? Comment briefly.

3.4 Computerized accounting

Beginning accounting students need to learn to learn to audit in a computerized environment. A sample of beginning accounting students took each of two tests: the Computer Attitude Scale (CAS) and the Computer Anxiety Rating Scale (CARS). A higher score in each indicates greater anxiety around computers. The test scores are scaled to be between 0 and 5. Also noted was each student's gender. The data are in http://ritsokiguess.site/datafiles/compatt.txt. The data values are separated by spaces.

(a) Read the data into R. Do you have what you expected? Explain briefly.

(b) How many males and females were there in the sample?

(c) Do the CAS scores tend to be higher for females or for males? Draw a suitable graph to help you decide, and come to a conclusion.

(d) Find the median CAS scores for each gender. Does this support what you saw on your plot? Explain briefly.

(e) Find the mean and standard deviation of both CAS and CARS scores (for all the students combined, ie. not separated by gender) *without* naming those columns explicitly.

3.5 Test scores in two classes

Open R Studio. Create a new Text File by selecting File, New File and Text File. You should see a new empty, untitled window appear at the top left. In that window, type or copy the data below (which are scores on a test for students in two different classes):

```
class score
ken 78
ken 62
ken 59
ken 69
ken 81
thomas 83
thomas 77
thomas 63
thomas 61
thomas 79
thomas 72
```

Save the file, using a filename of your choice (with, perhaps, extension `.txt`). Or, if you prefer, use the one at link.

(a) Read the data into a data frame called `marks`, using `read_delim`, and list the data frame (by typing its name) to confirm that you read the data values properly. Note that the top line of the data file contains the names of the variables, as it ought to.

(b) * Obtain side-by-side boxplots of the scores for each class.

(c) Do the two classes appear to have similar or different scores, on average? Explain briefly.

(d) Obtain a boxplot of all the scores together, regardless of which class they came from.

(e) Compute the median score (of all the scores together). Does this seem about right, looking at the boxplot? Explain briefly.

3.6 Unprecedented rainfall

In 1997, a company in Davis, California, had problems with odour in its wastewater facility. According to a company official, the problems were caused by "unprecedented weather conditions" and "because rainfall was at 170 to 180 percent of its normal level, the water in the holding ponds took longer to exit for irrigation, giving it more time to develop an odour."

Annual rainfall data for the Davis area is here. Note that clicking on the link will display the file, and *right*-clicking on the link will give you some options, one of which is Copy Link Address, which you can then paste into your R Notebook.

The rainfall is measured in inches.

(a) Read in and display (some of) the data.

(b) Summarize the data frame.

(c) Make a suitable plot of the rainfall values. (We are not, for the time being, concerned about the years.)

(d) How would you describe the shape of the distribution of rainfall values?

(e) In the quote at the beginning of the question, where do you think the assertion that the 1997 rainfall was "at 170 to 180 percent of its normal level" came from? Explain briefly.

(f) Do you think the official's calculation was reasonable? Explain briefly. (Note that this is not the same as asking whether the official's calculation was *correct*. This is an important distinction for you to make.)

(g) Do you think that the official was right to use the word "unprecedented" to describe the 1997 rainfall? Justify your answer briefly.

3.7 Learning algebra

At a high school in New Jersey, teachers were interested in what might help students to learn algebra. One idea was laptops as a learning aid, to see whether having access to one helped with algebra scores. (This was some time ago.) The 20 students in one class were given laptops to use in school and at home, while the 27 students in another class were not given laptops. For all of these students, the final exam score in algebra was recorded. The data are in http://ritsokiguess.site/datafiles/algebra.txt, with two columns, one indicating whether the student received a laptop or not, and the other giving their score on the algebra final exam.

(a) Read in and display (some of) the data. Do you have (i) the correct number of observations, and (ii) the correct *type* of columns? Explain briefly.

(b) Make a suitable graph of these data.

(c) Comment briefly on your graph, thinking about what the teachers would like to know.

(d) Work out the median and inter-quartile range for the students who did and who did not have laptops, and compare with the boxplot. (In R, the inter-quartile range is `IQR` in uppercase.)

My solutions follow:

3.8 North Carolina births

The data in file link are about 500 randomly chosen births of babies in North Carolina. There is a lot of information: not just the weight at birth of the baby, but whether the baby was born prematurely, the ages of the parents, whether the parents are married, how long (in weeks) the pregnancy lasted (this is called the "gestation") and so on.

(a) Read in the data from the file into R, bearing in mind what type of file it is.

Solution

This is a `.csv` file (it came from a spreadsheet), so it needs reading in accordingly. Work directly from the URL:

```
myurl <- "http://ritsokiguess.site/datafiles/ncbirths2.csv"
bw <- read_csv(myurl)
```

```
Rows: 500 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (10): father_age, mother_age, weeks_gestation, pre_natal_visits, marital...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

This shows you which variables the data set has (some of the names got a bit mangled), and it shows you that they are all integers except for the birth weight (a decimal number).

The easiest way to find out how many rows and columns there are is simply to list the data frame:

```r
bw
```

```
# A tibble: 500 x 10
   father_age mother_age weeks_gestation pre_natal_visits marital_status
        <dbl>      <dbl>           <dbl>            <dbl>          <dbl>
 1         27         26              38               14              1
 2         35         33              40               11              1
 3         34         22              37               10              2
 4         NA         16              38                9              2
 5         35         33              39               12              1
 6         32         24              36               12              1
 7         33         33              38               15              2
 8         38         35              38               16              1
 9         28         29              40                5              1
10         NA         19              34               10              2
# i 490 more rows
# i 5 more variables: mother_weight_gained <dbl>, low_birthweight <dbl>,
#   weight_pounds <dbl>, premie <dbl>, few_visits <dbl>
```

or you can take a "glimpse" of it, which is good if you have a lot of columns:

```r
glimpse(bw)
```

```
Rows: 500
Columns: 10
$ father_age           <dbl> 27, 35, 34, NA, 35, 32, 33, 38, 28, NA, 28, 34, N~
$ mother_age           <dbl> 26, 33, 22, 16, 33, 24, 33, 35, 29, 19, 26, 31, 1~
$ weeks_gestation      <dbl> 38, 40, 37, 38, 39, 36, 38, 38, 40, 34, 39, 39, 3~
$ pre_natal_visits     <dbl> 14, 11, 10, 9, 12, 12, 15, 16, 5, 10, 15, 15, 0, ~
$ marital_status       <dbl> 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2~
$ mother_weight_gained <dbl> 32, 23, 50, NA, 15, 12, 60, 2, 20, NA, 45, 22, 20~
$ low_birthweight      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
$ weight_pounds        <dbl> 6.8750, 6.8125, 7.2500, 8.8125, 8.8125, 5.8125, 6~
$ premie               <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0~
$ few_visits           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0~
```

Either of these displays show that there are 500 rows (observations, here births) and 10 columns (variables), and they both show what the variables are called. So they're both good as an answer to the question.

Extra: As is rather too often the way, the original data weren't like this, and I had to do some tidying first. Here's the original:

```
my_old_url <- "http://ritsokiguess.site/datafiles/ncbirths_original.csv"
bw0 <- read_csv(my_old_url)
```

```
Rows: 500 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (10): Father Age, Mother Age, Weeks Gestation, Pre-natal Visits, Marital...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bw0
```

```
# A tibble: 500 x 10
   `Father Age` `Mother Age` `Weeks Gestation` `Pre-natal Visits`
          <dbl>        <dbl>             <dbl>              <dbl>
 1           27           26                38                 14
 2           35           33                40                 11
 3           34           22                37                 10
 4           NA           16                38                  9
 5           35           33                39                 12
 6           32           24                36                 12
 7           33           33                38                 15
 8           38           35                38                 16
 9           28           29                40                  5
10           NA           19                34                 10
# i 490 more rows
# i 6 more variables: `Marital Status` <dbl>, `Mother Weight Gained` <dbl>,
#   `Low Birthweight?` <dbl>, `Weight (pounds)` <dbl>, `Premie?` <dbl>,
#   `Few Visits?` <dbl>
```

What you'll notice is that the variables have *spaces* in their names, which would require special handling later. The `glimpse` output shows you what to do about those spaces in variable names:

```
glimpse(bw0)
```

```
Rows: 500
Columns: 10
$ `Father Age`           <dbl> 27, 35, 34, NA, 35, 32, 33, 38, 28, NA, 28, 34,~
$ `Mother Age`           <dbl> 26, 33, 22, 16, 33, 24, 33, 35, 29, 19, 26, 31,~
$ `Weeks Gestation`      <dbl> 38, 40, 37, 38, 39, 36, 38, 38, 40, 34, 39, 39,~
$ `Pre-natal Visits`     <dbl> 14, 11, 10, 9, 12, 12, 15, 16, 5, 10, 15, 15, 0~
$ `Marital Status`       <dbl> 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 2, 2,~
$ `Mother Weight Gained` <dbl> 32, 23, 50, NA, 15, 12, 60, 2, 20, NA, 45, 22, ~
$ `Low Birthweight?`     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,~
$ `Weight (pounds)`      <dbl> 6.8750, 6.8125, 7.2500, 8.8125, 8.8125, 5.8125,~
$ `Premie?`              <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,~
$ `Few Visits?`          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,~
```

What you have to do is to surround the variable name with "backticks". (On my keyboard, that's on the key to the left of number 1, where the squiggle is, that looks like a backwards apostrophe. Probably next to `Esc`, depending on the layout of your keyboard.) For example, to get the mean mother's age, you have to do this:

```r
bw0 %>% summarize(mom_mean = mean(`Mother Age`))
```

```
# A tibble: 1 x 1
  mom_mean
     <dbl>
1     26.9
```

Although almost all of the variables are stored as integers, the ones that have a question mark in their name are actually "logical", true or false, with 1 denoting true and 0 false. We could convert them later if we want to. A question mark is not a traditional character to put in a variable name either, so we have to surround these variables with backticks too.

In fact, all the variables have "illegal" names in one way or another: they contain spaces, or question marks, or brackets. So they *all* need backticks, which, as you can imagine, is rather awkward. The Capital Letters at the start of each word are also rather annoying to type every time.

People who collect data are not always the people who analyze it, so there is not always a lot of thought given to column names in spreadsheets.

So how did I get you a dataset with much more sane variable names? Well, I used the `janitor` package, which has a function in it called `clean_names`. This is what it does:

```r
library(janitor)
bw0 %>% clean_names() %>% glimpse()
```

```
Rows: 500
Columns: 10
$ father_age           <dbl> 27, 35, 34, NA, 35, 32, 33, 38, 28, NA, 28, 34, N~
$ mother_age           <dbl> 26, 33, 22, 16, 33, 24, 33, 35, 29, 19, 26, 31, 1~
$ weeks_gestation      <dbl> 38, 40, 37, 38, 39, 36, 38, 38, 40, 34, 39, 39, 3~
$ pre_natal_visits     <dbl> 14, 11, 10, 9, 12, 12, 15, 16, 5, 10, 15, 15, 0, ~
$ marital_status       <dbl> 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2~
$ mother_weight_gained <dbl> 32, 23, 50, NA, 15, 12, 60, 2, 20, NA, 45, 22, 20~
$ low_birthweight      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
$ weight_pounds        <dbl> 6.8750, 6.8125, 7.2500, 8.8125, 8.8125, 5.8125, 6~
$ premie               <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0~
$ few_visits           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0~
```

All the spaces have been replaced by underscores, the question marks and brackets have been removed, and all the uppercase letters have been made lowercase. The spaces have been replaced by underscores because an underscore is a perfectly legal thing to have in a variable name. I saved this dataset into the file you read in.

■

(b) From your output, verify that you have the right number of observations and that you have several variables. Which of your variables correspond to birthweight, prematureness and length of pregnancy? (You might have to make guesses based on the names of the variables.)

Solution

As a reminder:

```r
glimpse(bw)
```

```
Rows: 500
Columns: 10
$ father_age           <dbl> 27, 35, 34, NA, 35, 32, 33, 38, 28, NA, 28, 34, N~
$ mother_age           <dbl> 26, 33, 22, 16, 33, 24, 33, 35, 29, 19, 26, 31, 1~
$ weeks_gestation      <dbl> 38, 40, 37, 38, 39, 36, 38, 38, 40, 34, 39, 39, 3~
$ pre_natal_visits     <dbl> 14, 11, 10, 9, 12, 12, 15, 16, 5, 10, 15, 15, 0, ~
$ marital_status       <dbl> 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2~
$ mother_weight_gained <dbl> 32, 23, 50, NA, 15, 12, 60, 2, 20, NA, 45, 22, 20~
$ low_birthweight      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
$ weight_pounds        <dbl> 6.8750, 6.8125, 7.2500, 8.8125, 8.8125, 5.8125, 6~
$ premie               <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0~
$ few_visits           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0~
```

I do indeed have 500 observations (rows) on 10 variables (columns; "several"). (If you don't have several variables, check to see that you didn't use `read_delim` or something by mistake.) After that, you see all the variables by name, with what type of values they have,[1] and the first few of the values.[2]

The variable `weight_pounds` is the birthweight (in pounds), `premie` is 1 for a premature baby and 0 for a full-term baby, and `weeks_gestation` is the number of weeks the pregnancy lasted.

■

(c) The theory behind the t-test (which we do later) says that the birth weights should be (approximately) normally distributed. Obtain a histogram of the birth weights. Does it look approximately normal? Comment briefly. (You'll have to pick a number of bins for your histogram first. I don't mind very much what you pick, as long as it's not obviously too many or too few bins.)

Solution

You'll have seen that I often start with 10 bins, or maybe not quite that many if I don't have much data, and this is a decent general principle. That would give

```
ggplot(bw, aes(x = weight_pounds)) + geom_histogram(bins = 10)
```

[1] these are mostly `int`, that is, integer.

[2] Other possible variable types are *num* for (real, decimal) numbers such as birth weight, *chr* for text, and *Factor* (with the number of levels) for factors/categorical variables. We don't have any of the last two here. There is also *lgl* for *logical*, things that were actually recorded as TRUE or FALSE. We have some variables that are actually logical ones, but they are recorded as integer values.

which is perfectly acceptable with 500 observations. You can try something a bit more or a bit less, and see how you like it in comparison. What you are looking for is a nice clear picture of *shape*. If you have too few bins, you'll lose the shape:

```
ggplot(bw, aes(x = weight_pounds)) + geom_histogram(bins = 4)
```

(is that leftmost bin an indication of skewness or some observations that happen to be small-ish?)

And if you have too many, the shape will be there, but it will be hard to make out in all the noise, with frequencies going up and down:

```
ggplot(bw, aes(x = weight_pounds)) + geom_histogram(bins = 30)
```

I generally am fairly relaxed about the number of bins you use, as long as it's not clearly too few or too many. You might have done exercises in the past that illustrate that the choice of number of bins (or the class intervals where you move from one bin to the next, which is another issue that I won't explore here) can make an appreciable difference to how a histogram looks.

Extra: I had some thoughts about this issue that I put in a blog post, that you might like to read: link. The nice thing about Sturges' rule, mentioned there, is that you can almost get a number of bins for your histogram in your head (as long as you know the powers of 2, that is). What you do is to start with your sample size, here $n = 500$. You find the next power of 2 above that, which is here $512 = 2^9$. You then take that power and add 1, to get 10 bins. If you don't like that, you can get R to calculate it for you:

```
nclass.Sturges(bw$weight_pounds)
```

```
[1] 10
```

The place where Sturges' rule comes from is an assumption of normal data (actually a binomial approximation to the normal, backwards though that sounds). If you have less than 30 observations, you'll get fewer than 6 bins, which won't do much of a job of showing the shape. Rob Hyndman wrote a critical note about Sturges' rule in which he asserts that it is just plain wrong (if you have taken B57, this note is very readable).

So what to use instead? Well, judgment is still better than something automatic, but if you want a place to start from, something with a better foundation than Sturges is the Freedman-Diaconis rule. This, in its original formulation, gives a bin width rather than a number of bins:

$$w = 2(IQR)n^{-1/3}$$

The nice thing about this is that it uses the interquartile range, so it won't be distorted by outliers. `geom_histogram` can take a bin width, so we can use it as follows:

```
w <- 2 * IQR(bw$weight_pounds) * 500^(-1 / 3)
w
```

```
[1] 0.4094743
```

```
ggplot(bw, aes(x = weight_pounds)) + geom_histogram(binwidth = w)
```

R also has

```
nc <- nclass.FD(bw$weight_pounds)
nc
```

```
[1] 26
```

which turns the Freedman-Diaconis rule into a number of bins rather than a binwidth; using that gives the same histogram as we got with `binwidth`.

In my opinion, Freedman-Diaconis tends to give too many bins (here there are 26 rather than the 10 of Sturges). But I put it out there for you to make your own call.

Another way to go is a "density plot". This is a smoothed-out version of a histogram that is not obviously frequencies in bins, but which does have a theoretical basis. It goes something like this:

```
ggplot(bw, aes(x = weight_pounds)) + geom_density()
```

`geom_density` has an optional parameter that controls how smooth or wiggly the picture is, but the default is usually good.

Alright, before we got distracted, we were assessing normality. What about that?

It is mostly normal-looking, but I am suspicious about those *very* low birth weights, the ones below about 4 pounds. There are a few too many of those, as I see it.

If you think this is approximately normal, you need to make some comment along the lines of "the shape is approximately symmetric with no outliers". I think my first answer is better,

but this answer is worth something, since it is a not completely unreasonable interpretation of the histogram.

I have been making the distinction between a histogram (for one quantitative variable) and side-by-side boxplots (for one quantitative variable divided into groups by one categorical variable). When you learned the boxplot, you probably learned it in the context of one quantitative variable. You can draw a boxplot for that, too, but the `ggplot` boxplot has an `x` as well as a `y`. What you do to make a single boxplot is to set the `x` equal 1, which produces a weird x-axis (that you ignore):

```
ggplot(bw, aes(x = 1, y = weight_pounds)) + geom_boxplot()
```

The high weight is actually an outlier, but look at all those outliers at the bottom![3]

I think the reason for those extra very low values is that they are the premature births (that can result in *very* small babies). Which leads to the additional question coming up later.

■

[3]When Tukey, a name we will see again, invented the boxplot in the 1950s, 500 observations would have been considered a big data set. He designed the boxplot to produce a sensible number of outliers for the typical size of data set of his day, but a boxplot of a large data set tends to have a lot of outliers that are probably not really outliers at all.

3.9 More about the NC births

This is an exploration of some extra issues around the North Carolina births data set.

(a) How short does a pregnancy have to be, for the birth to be classified as "premature"? Deduce this from the data, by drawing a suitable graph or otherwise.

Solution

To figure it out from the data, we can see how `weeks_gestation` depends on `premie`. Some possibilities are boxplots or a scatterplot. Either of the first two graphs would get full credit (for the graphing part: you still have to do the explanation) if this were being marked:

```
ggplot(bw,aes(x=factor(premie), y=weeks_gestation)) + geom_boxplot()
```

```
Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```

The warning is because the prematurity of one of the babies is not known. Or

```
ggplot(bw,aes(x=premie, y=weeks_gestation)) + geom_point()
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
```

The same warning again, for the same reason.

Notice how the graphs are similar in syntax, because the what-to-plot is the same (apart from the `factor` thing) and we just make a small change in how-to-plot-it. In the boxplot, the thing on the x-scale needs to be categorical, and `premie` is actually a number, so we'd better make it into a `factor`, which is R's version of a categorical variable. `premie` *is* actually a categorical variable ("premature" or "not premature") masquerading as a quantitative one (1 or 0). It is an "indicator variable", if you're familiar with that term.

It looks as if the breakpoint is 37 weeks: a pregnancy at least that long is considered normal, but a shorter one ends with a premature birth. Both plots show the same thing: the `premie=1` births all go with short pregnancies, shorter than 37 weeks. This is completely clear cut.

Another way to attack this is to use `summarize`, finding the max and min:

```
bw %>% summarize( n=n(),
min=min(weeks_gestation),
max=max(weeks_gestation))
```

```
# A tibble: 1 x 3
      n   min   max
  <int> <dbl> <dbl>
1   500    NA    NA
```

only this is for *all* the babies, premature or not.[4] So we want it by prematurity, which means a `group_by` first:

```
bw %>% group_by(premie) %>%
summarize( n=n(),
min=min(weeks_gestation),
max=max(weeks_gestation))
```

```
# A tibble: 3 x 4
  premie     n   min   max
   <dbl> <int> <dbl> <dbl>
1      0   424    37    45
2      1    75    20    36
3     NA     1    NA    NA
```

`group_by` with a number works, even though using the number in `premie` in a boxplot didn't. `group_by` just uses the distinct values, whether they are numbers, text or factor levels.

Any of these graphs or summaries will help you answer the question, in the same way. The ultimate issue here is "something that will get the job done": it doesn't matter so much what.

Extra: In R, `NA` means "missing". When you try to compute something containing a missing value, the answer is usually missing (since you don't know what the missing value is). That's why the first `summarize` gave us missing values: there was one missing weeks of gestation in with all the ones for which we had values, so the max and min had to be missing as well. In the second `summarize`, the one by whether a baby was born prematurely or not, we learn a bit more about that missing `premie`: evidently its weeks of gestation was missing as well, since the min and max of that were missing.[5]

Here's that baby. I'm doing a bit of fiddling to show all the columns (as rows, since there's only one actual row). Don't worry about the second line of code below; we will investigate that later in the course. Its job here is to show the values nicely:

```
bw %>%
  filter(is.na(premie)) %>%
  pivot_longer(everything(), names_to="name", values_to="value")
```

[4] I explain the missing values below.

[5] If there had been a weeks of gestation, we could have figured out whether it was premature or not, according to whether the weeks of gestation was less than 37.

```
# A tibble: 10 x 2
   name                 value
   <chr>                <dbl>
 1 father_age           33
 2 mother_age           32
 3 weeks_gestation      NA
 4 pre_natal_visits      9
 5 marital_status        1
 6 mother_weight_gained 25
 7 low_birthweight       0
 8 weight_pounds         7.19
 9 premie               NA
10 few_visits            0
```

The *only* thing that was missing was its weeks of gestation, but that prevented anyone from figuring out whether it was premature or not.

■

(b) Explore the relationship between birth weight and length of pregancy ("gestation") using a suitable graph. What do you see?

Solution

This needs to be a scatterplot because these are both quantitative variables:

```r
ggplot(bw,aes(x=weeks_gestation, y=weight_pounds)) + geom_point()
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
```

You see a rather clear upward trend. Those very underweight babies came from very short pregnancies, but the vast majority of pregnancies were of more or less normal length (40 weeks is normal) and resulted in babies of more or less normal birth weight.

Extra: I want to illustrate something else: how about *colouring* the births that were premature? Piece of cake with `ggplot`:

```
ggplot(bw,aes(x=weeks_gestation, y=weight_pounds, colour = premie)) +
  geom_point()
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
```

That was rather silly because `ggplot` treated prematureness as a *continuous* variable, and plotted the values on a dark blue-light blue scale. This is the same issue as on the boxplot above, and has the same solution:

```
ggplot(bw,aes(x=weeks_gestation, y=weight_pounds,
              colour = factor(premie))) + geom_point()
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
```

Better.

With the normal-length pregnancies (red), there seems to be no relationship between length of pregnancy and birth weight, just a random variation. But with the premature births, a shorter pregnancy typically goes with a *lower* birth weight. This would be why the birth weights for the premature births were more variable.

■

(c) Do a web search to find the standard (North American) definition of a premature birth. Does that correspond to what you saw in the data? Cite the website you used, for example by saying "according to `URL`, …", with `URL` replaced by the address of the website you found.

Solution

The website http://www.mayoclinic.org/diseases-conditions/premature-birth/basics/definition/con-20020050 says that "a premature birth is one that occurs before the start of the 37th week of pregnancy", which is exactly what we found. (Note that I am citing the webpage on which I found this, and I even made it into a link so that you can check it.) The Mayo Clinic is a famous hospital system with locations in several US states, so I think we can trust what its website says.

■

3.10 Nenana, Alaska

Nenana, Alaska, is about 50 miles west of Fairbanks. Every spring, there is a contest in Nenana. A wooden tripod is placed on the frozen river, and people try to guess the exact minute when the ice melts enough for the tripod to fall through the ice. The contest started in 1917 as an amusement for railway workers, and has taken place every year since. Now, hundreds of thousands of people enter their guesses on the Internet and the prize for the winner can be as much as $300,000.

Because so much money is at stake, and because the exact same tripod is placed at the exact same spot on the ice every year, the data are consistent and accurate. The data are in link.

(a) Read the data into R. Note that the values are separated by *tabs* rather than spaces, so you'll need an appropriate `read_` to read it in.

Solution

These are "tab-separated values", so `read_tsv` is the thing, as for the Australian athletes:

```
myurl <- "http://ritsokiguess.site/datafiles/nenana.txt"
nenana <- read_tsv(myurl)
```

```
Rows: 87 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): Date&Time
dbl (2): Year, JulianDate

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Use whatever name you like for the data frame. One that is different from any of the column headers is smart; then it is clear whether you mean the whole data frame or one of its columns. `ice` or `melt` or anything like that would also be good.

I haven't asked you to display or check the data (that's coming up), but if you look at it and find that it didn't work, you'll know to come back and try this part again. R usually gets it right or gives you an error.

If you look at the data, they do appear to be separated by spaces, but the text version of the date and time *also* have spaces in them, so things might go astray if you try and read the values in without recognizing that the actual separator is a tab:

```
x <- read_delim(myurl, " ")
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for details,
e.g.:
  dat <- vroom(...)
  problems(dat)

Rows: 87 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): Year   JulianDate  Date&Time

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Ouch! A hint as to what went wrong comes from looking at the read-in data frame:

```r
x
```

```
# A tibble: 87 x 1
   `Year\tJulianDate\tDate&Time`
   <chr>
 1 "1917\t120.4795\tApril 30 at 11:30 AM"
 2 "1918\t131.3983\tMay 11 at 9:33 AM"
 3 "1919\t123.6066\tMay 3 at 2:33 PM"
 4 "1920\t132.4490\tMay 11 at 10:46 AM"
 5 "1921\t131.2795\tMay 11 at 6:42 AM"
 6 "1922\t132.5559\tMay 12 at 1:20 PM"
 7 "1923\t129.0837\tMay 9 at 2:00 AM"
 8 "1924\t132.6323\tMay 11 at 3:10 PM"
 9 "1925\t127.7726\tMay 7 at 6:32 PM"
10 "1926\t116.6691\tApril 26 at 4:03 PM"
# i 77 more rows
```

Those `t` symbols mean "tab character", which is our hint that the values were separated by tabs rather than spaces.

More detail (if you can bear to see it) is here:

```r
problems(x)
```

```
# A tibble: 87 x 5
     row   col expected  actual    file
```

```
   <int> <int> <chr>     <chr>     <chr>
 1     2     5 1 columns 5 columns ""
 2     3     5 1 columns 5 columns ""
 3     4     5 1 columns 5 columns ""
 4     5     5 1 columns 5 columns ""
 5     6     5 1 columns 5 columns ""
 6     7     5 1 columns 5 columns ""
 7     8     5 1 columns 5 columns ""
 8     9     5 1 columns 5 columns ""
 9    10     5 1 columns 5 columns ""
10    11     5 1 columns 5 columns ""
# i 77 more rows
```

The first line of the data file (with the variable names in it) had no spaces, only tabs, so `read_delim` thinks there is *one* column with a very long name, but in the actual data, there are *five* space-separated columns. The text date-times are of the form "April 30 at 11:30 AM", which, if you think it's all separated by spaces, is actually 5 things: April, 30, at and so on. These are the only things that are separated by spaces, so, from that point of view, there are five columns.

■

(b) Find a way of displaying how many rows and columns your data frame has, and some of the values. Describe the first and last of the variables that you appear to have.

Solution

The easiest is just to display the tibble:

```r
nenana
```

```
# A tibble: 87 x 3
    Year JulianDate `Date&Time`
   <dbl>      <dbl> <chr>
 1  1917       120. April 30 at 11:30 AM
 2  1918       131. May 11 at 9:33 AM
 3  1919       124. May 3 at 2:33 PM
 4  1920       132. May 11 at 10:46 AM
 5  1921       131. May 11 at 6:42 AM
 6  1922       133. May 12 at 1:20 PM
 7  1923       129. May 9 at 2:00 AM
 8  1924       133. May 11 at 3:10 PM
 9  1925       128. May 7 at 6:32 PM
```

```
10  1926       117. April 26 at 4:03 PM
# i 77 more rows
```

Alternatively, you can take a `glimpse` of it:

```r
glimpse(nenana)
```

```
Rows: 87
Columns: 3
$ Year        <dbl> 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926~
$ JulianDate  <dbl> 120.4795, 131.3983, 123.6066, 132.4490, 131.2795, 132.5559~
$ `Date&Time` <chr> "April 30 at 11:30 AM", "May 11 at 9:33 AM", "May 3 at 2:3~
```

There are 87 years, and 3 columns (variables). The first column is year, and the last column is the date and time that the tripod fell into the river, written as a piece of text. I explain the second column in a moment.

■

(c) Dates and times are awkward to handle with software. (We see more ways later in the course.) The column `JulianDate` expresses the time that the tripod fell through the ice as a fractional number of days since December 31. This enables the time (as a fraction of the way through the day) to be recorded as well, the whole thing being an ordinary number. Make a histogram of the Julian dates. Comment briefly on its shape.

Solution

With a `ggplot` histogram, we need a number of bins first. I can do Sturges' rule in my head: the next power of 2 up from 87 (our n) is 128, which is 2^7, so the base 2 log of 87 rounds up to 7. That plus one is 8, so we need 8 bins. For you, any not-insane number of bins will do, or any not-insane bin width, if you want to go that way:

```r
ggplot(nenana, aes(x = JulianDate)) + geom_histogram(bins = 8)
```

Note that you need to type `JulianDate` exactly as it appears, capital letters and all. R is case-sensitive.

This histogram looks more or less symmetric (and, indeed, normal). I really don't think you can justify an answer other than "symmetric" here. (Or "approximately normal": that's good too.) If your histogram is different, say so. I think that "hole" in the middle is not especially important.

We haven't done normal quantile plots yet, but looking ahead:

```
ggplot(nenana, aes(sample = JulianDate)) + stat_qq() + stat_qq_line()
```

That hugs the line pretty well, so I would call it close to normally-distributed. It bulges away from the line because there are more values just below 120 than you would expect for a normal. This corresponds to the histogram bar centred just below 120 being taller than you would have expected.[6]

Extra: looking *way* ahead (to almost the end of the R stuff), this is how you handle the dates and times:

```
library(lubridate)
nenana %>%
  mutate(longdt = str_c(Year, " ", `Date&Time`)) %>%
  mutate(datetime = ymd_hm(longdt, tz = "America/Anchorage"))
```

```
# A tibble: 87 x 5
    Year JulianDate `Date&Time`          longdt              datetime
   <dbl>      <dbl> <chr>                <chr>               <dttm>
 1  1917       120. April 30 at 11:30 AM 1917 April 30 at 1~ 1917-04-30 11:30:00
 2  1918       131. May 11 at 9:33 AM    1918 May 11 at 9:3~ 1918-05-11 09:33:00
 3  1919       124. May 3 at 2:33 PM     1919 May 3 at 2:33~ 1919-05-03 14:33:00
 4  1920       132. May 11 at 10:46 AM   1920 May 11 at 10:~ 1920-05-11 10:46:00
 5  1921       131. May 11 at 6:42 AM    1921 May 11 at 6:4~ 1921-05-11 06:42:00
```

[6]That is to say, the principal deviation from normality is not the hole on the histogram, the bar centred around 123 being too short, but that the bar centred just below 120 is too *tall*.

```
 6  1922       133. May 12 at 1:20 PM    1922 May 12 at 1:2~ 1922-05-12 13:20:00
 7  1923       129. May 9 at 2:00 AM     1923 May 9 at 2:00~ 1923-05-09 02:00:00
 8  1924       133. May 11 at 3:10 PM    1924 May 11 at 3:1~ 1924-05-11 15:10:00
 9  1925       128. May 7 at 6:32 PM     1925 May 7 at 6:32~ 1925-05-07 18:32:00
10  1926       117. April 26 at 4:03 PM  1926 April 26 at 4~ 1926-04-26 16:03:00
# i 77 more rows
```

I am not doing any further analysis with these, so just displaying them is good.

I have to do a preliminary step to get the date-times *with* their year in one place. `str_c` glues pieces of text together: in this case, the year, a space, and then the rest of the `Date&Time`. I stored this in `longdt`. The second `mutate` is the business end of it: `ymd_hm` takes a piece of text containing a year, month (by name or number), day, hours, minutes *in that order*, and extracts those things from it, storing the whole thing as an R date-time. Note that the AM/PM was handled properly. The benefit of doing that is we can extract anything from the dates, such as the month or day of week, or take differences between the dates. Or even check that the Julian dates were calculated correctly (the `lubridate` function is called `yday` for "day of year"):

```r
nenana %>%
  mutate(longdt = str_c(Year, " ", `Date&Time`)) %>%
  mutate(datetime = ymd_hm(longdt, tz = "America/Anchorage")) %>%
  mutate(jd = yday(datetime)) ->
nenana2
nenana2 %>% select(JulianDate, jd, datetime)
```

```
# A tibble: 87 x 3
   JulianDate    jd datetime
        <dbl> <dbl> <dttm>
 1       120.   120 1917-04-30 11:30:00
 2       131.   131 1918-05-11 09:33:00
 3       124.   123 1919-05-03 14:33:00
 4       132.   132 1920-05-11 10:46:00
 5       131.   131 1921-05-11 06:42:00
 6       133.   132 1922-05-12 13:20:00
 7       129.   129 1923-05-09 02:00:00
 8       133.   132 1924-05-11 15:10:00
 9       128.   127 1925-05-07 18:32:00
10       117.   116 1926-04-26 16:03:00
# i 77 more rows
```

The Julian days as calculated are the same. Note that these are not rounded; the Julian day begins at midnight and lasts until the next midnight. Thus Julian day 132 is May 12 (in a

non-leap year like 1922) and the reason that the Julian date given in the file for that year would round to 133 is that it is after noon (1:20pm as you see).

■

(d) Plot `JulianDate` against `Year` on a scatterplot. What recent trends, if any, do you see? Comment briefly.

Solution

`geom_point`:

```
ggplot(nenana, aes(x = Year, y = JulianDate)) + geom_point()
```

This is actually a small-but-real downward trend, especially since about 1960, but the large amount of variability makes it hard to see, so I'm good with either "no trend" or "weak downward trend" or anything roughly like that. There is definitely not much trend before 1960, but most of the really early break-ups (less than about 118) have been since about 1990.

You can even add to the `ggplot`, by putting a smooth trend on it:

```
ggplot(nenana, aes(x = Year, y = JulianDate)) + geom_point() + geom_smooth()
```

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

This is R's version of a trend that is not constrained to be linear (so that it "lets the data speak for itself").

Now there is something obvious to see: after about 1960, there is a clear downward trend: the ice is breaking up earlier on average every year. Even though there is a lot of variability, the overall trend, viewed this way, is clear.

What does this mean, in practice? This notion of the ice melting earlier than it used to is consistent all over the Arctic, and is one more indication of climate change. Precisely, it is an indication that climate change is happening, but we would have to delve further to make any statements about the *cause* of that climate change.

■

3.11 Computerized accounting

Beginning accounting students need to learn to learn to audit in a computerized environment. A sample of beginning accounting students took each of two tests: the Computer Attitude Scale (CAS) and the Computer Anxiety Rating Scale (CARS). A higher score in each indicates greater anxiety around computers. The test scores are scaled to be between 0 and 5. Also noted

was each student's gender. The data are in http://ritsokiguess.site/datafiles/compatt.txt. The data values are separated by spaces.

(a) Read the data into R. Do you have what you expected? Explain briefly.

Solution

Read in and display the data. This, I think, is the easiest way.

```
my_url <- "https://raw.githubusercontent.com/nxskok/datafiles/master/compatt.txt"
anxiety=read_delim(my_url," ")
```

```
Rows: 35 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): gender
dbl (2): CAS, CARS

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
anxiety
```

```
# A tibble: 35 x 3
   gender   CAS  CARS
   <chr>  <dbl> <dbl>
 1 female  2.85  2.9
 2 male    2.6   2.32
 3 female  2.2   1
 4 male    2.65  2.58
 5 male    2.6   2.58
 6 male    3.2   3.05
 7 male    3.65  3.74
 8 female  2.55  1.9
 9 male    3.15  3.32
10 male    2.8   2.74
# i 25 more rows
```

There is a total of 35 students with a CAS score, a CARS score and a gender recorded for each. This is in line with what I was expecting. (You can also note that the genders appear to be a mixture of males and females.)

■

(b) How many males and females were there in the sample?

Solution

Most easily `count`:

```
anxiety %>% count(gender)
```

```
# A tibble: 2 x 2
  gender     n
  <chr>  <int>
1 female    15
2 male      20
```

This also works (and is therefore good):

```
anxiety %>% group_by(gender) %>% summarize(count=n())
```

```
# A tibble: 2 x 2
  gender count
  <chr>  <int>
1 female    15
2 male      20
```

I want you to use R to do the counting (that is, don't just go through the whole data set and count the males and females yourself). This is because you might have thousands of data values and you need to learn how to get R to count them for you.

15 females and 20 males, *which you should say*. I made a point of *not* saying that it is enough to get the output with the answers on it, so you need to tell me what the answer is.

■

(c) Do the CAS scores tend to be higher for females or for males? Draw a suitable graph to help you decide, and come to a conclusion.

Solution

Gender is categorical and CAS score is quantitative, so a boxplot would appear to be the thing:

```
ggplot(anxiety,aes(x=gender,y=CAS))+geom_boxplot()
```

The median for males is slightly higher, so male accountants are more anxious around computers than female accountants are.

If you wish, side-by-side (or, better, above-and-below) histograms would also work:

```
ggplot(anxiety,aes(x=CAS))+geom_histogram(bins=6)+
facet_wrap(~gender,ncol=1)
```

If you go this way, you have to make a call about where the centres of the histograms are. I guess the male one is slightly further to the right, but it's not so easy to tell. (Make a call.)

■

(d) Find the median CAS scores for each gender. Does this support what you saw on your plot? Explain briefly.

Solution

Group-by and summarize:

```
anxiety %>% group_by(gender) %>% summarize(med=median(CAS))
```

```
# A tibble: 2 x 2
  gender   med
  <chr>  <dbl>
1 female  2.55
2 male    2.85
```

The median is a bit higher for males, which is what I got on my boxplot (and is apparently the same thing as is on the histograms, but it's harder to be sure there).

■

(e) Find the mean and standard deviation of both CAS and CARS scores (for all the students combined, ie. not separated by gender) *without* naming those columns explicitly.

Solution

Without naming them explicitly means using some other way to pick them out of the data frame, `summarize` with `across`.

The basic `across` comes from asking yourself what the *names* of those columns have in common: they start with C and the gender column doesn't:

```
anxiety %>% summarize(across(starts_with("C"), list(m = ~mean(.), s = ~sd(.))))
```

```
# A tibble: 1 x 4
  CAS_m CAS_s CARS_m CARS_s
  <dbl> <dbl>  <dbl>  <dbl>
1  2.82 0.484   2.77  0.671
```

Another way is to ask what *property* these two columns have in common: they are the only two numeric (quantitative) columns. This means using an `across` with a `where` inside it, thus:

```
anxiety %>% summarize(across(where(is.numeric), list(m = ~mean(.), s = ~sd(.))))
```

```
# A tibble: 1 x 4
  CAS_m CAS_s CARS_m CARS_s
  <dbl> <dbl>  <dbl>  <dbl>
1  2.82 0.484   2.77  0.671
```

Read the first one as "across all the columnns whose names start with S, find the mean and SD of them." The second one is a little clunkier: "acrosss all the columns for which `is.numeric` is true, find the mean and SD of them". A shorter way for the second one is "across all the numeric (quantitative) columns, find their mean and SD", but then you have to remember exactly how to code that. The reason for the `list` is that we are calculating two statistics for each column that we find. I am using a "named list" so that the mean gets labelled with an `m` on the end of the column name, and the SD gets an `s` on the end.

Either of these is good, or anything equivalent (like noting that the two anxiety scales both `ends\_with` S):

```
anxiety %>% summarize(across(ends_with("S"), list(m = ~mean(.), s = ~sd(.))))
```

```
# A tibble: 1 x 4
  CAS_m CAS_s CARS_m CARS_s
  <dbl> <dbl>  <dbl>  <dbl>
1  2.82 0.484   2.77  0.671
```

Because I didn't say otherwise, you should tell me what the means and SDs are, rounding off suitably: the CAS scores have mean 2.82 and SD 0.48, and the CARS scores have mean 2.77 and SD 0.67.

Yet another way to do it is to select the columns you want first (which you can do by number so as not to name them), and then find the mean and SD of all of them:

```
anxiety %>% select(2:3) %>%
    summarize(across(everything(), list(m = ~mean(.), s = ~sd(.))))
```

```
# A tibble: 1 x 4
  CAS_m CAS_s CARS_m CARS_s
  <dbl> <dbl>  <dbl>  <dbl>
1  2.82 0.484   2.77  0.671
```

This doesn't work:

```
summary(anxiety)
```

```
    gender               CAS             CARS      
 Length:35          Min.   :1.800   Min.   :1.000  
 Class :character   1st Qu.:2.575   1st Qu.:2.445  
 Mode  :character   Median :2.800   Median :2.790  
                    Mean   :2.816   Mean   :2.771  
                    3rd Qu.:3.150   3rd Qu.:3.290  
                    Max.   :3.750   Max.   :4.000  
```

because, although it gets the means, it does not get the standard deviations. (I added the SD to the original question to make you find a way other than this.)

In summary, find a way to get those answers without naming those columns in your code, and I'm good.

In case you were wondering about how to do this separately by gender, well, put the `group\_by` in like you did before:

```r
anxiety %>% group_by(gender) %>%
summarize(across(where(is.numeric), list(m = ~mean(.), s = ~sd(.))))
```

```
# A tibble: 2 x 5
  gender CAS_m CAS_s CARS_m CARS_s
  <chr>  <dbl> <dbl>  <dbl>  <dbl>
1 female  2.64 0.554   2.51  0.773
2 male    2.94 0.390   2.96  0.525
```

or

```r
anxiety %>% group_by(gender) %>%
summarize(across(starts_with("C"), list(m = ~mean(.), s = ~sd(.))))
```

```
# A tibble: 2 x 5
  gender CAS_m CAS_s CARS_m CARS_s
  <chr>  <dbl> <dbl>  <dbl>  <dbl>
1 female  2.64 0.554   2.51  0.773
2 male    2.94 0.390   2.96  0.525
```

The male means are slightly higher on both tests, but the male standard deviations are a little smaller. You might be wondering whether the test scores are related. They are both quantitative, so the obvious way to find out is a scatterplot:

```r
ggplot(anxiety,aes(x=CAS,y=CARS))+geom_point()
```

The two variables can be on either axis, since there is no obvious response or explanatory variable. A higher score on one scale goes with a higher score on the other, suggesting that the two scales are measuring the same thing.

This plot mixes up the males and females, so you might like to distinguish them, which goes like this:

```
ggplot(anxiety,aes(x=CAS,y=CARS,colour=gender))+geom_point()
```

There is a slight (but only slight) tendency for the males to be up and to the right, and for the females to be down and to the left. This is about what you would expect, given that the male means are slightly bigger on both scores, but the difference in means is not that big compared to the SD.

■

3.12 Test scores in two classes

Open R Studio. Create a new Text File by selecting File, New File and Text File. You should see a new empty, untitled window appear at the top left. In that window, type or copy the data below (which are scores on a test for students in two different classes):

```
class score
ken 78
ken 62
ken 59
ken 69
ken 81
thomas 83
thomas 77
```

```
thomas 63
thomas 61
thomas 79
thomas 72
```

Save the file, using a filename of your choice (with, perhaps, extension `.txt`). Or, if you prefer, use the one at link.

(a) Read the data into a data frame called `marks`, using `read_delim`, and list the data frame (by typing its name) to confirm that you read the data values properly. Note that the top line of the data file contains the names of the variables, as it ought to.

Solution

I was lazy and used the one on the web, the values being separated ("delimited") by exactly one space:

```r
my_url <- "http://ritsokiguess.site/datafiles/marks.txt"
marks <- read_delim(my_url, " ")
```

```
Rows: 11 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): class
dbl (1): score

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
marks
```

```
# A tibble: 11 x 2
   class  score
   <chr>  <dbl>
 1 ken       78
 2 ken       62
 3 ken       59
 4 ken       69
 5 ken       81
 6 thomas    83
 7 thomas    77
```

```
 8 thomas    63
 9 thomas    61
10 thomas    79
11 thomas    72
```

If you copied and pasted, or typed in, the data values yourself, use the local file name (such as `marks.txt`) in place of the URL.

Extra: in the old days, when we used `read.table` (which actually also works here), we needed to also say `header=T` to note that the top line of the data file was variable names. With `read_delim`, that's the default, and if the top line is *not* variable names, that's when you have to say so. If I cheat, by skipping the first line and saying that I then have no column names, I get:

```
read_delim(my_url, " ", col_names = F, skip = 1)
```

```
Rows: 11 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): X1
dbl (1): X2

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

# A tibble: 11 x 2
   X1        X2
   <chr>  <dbl>
 1 ken       78
 2 ken       62
 3 ken       59
 4 ken       69
 5 ken       81
 6 thomas    83
 7 thomas    77
 8 thomas    63
 9 thomas    61
10 thomas    79
11 thomas    72
```

Column names are supplied (`X1` and `X2`). I could also supply my own column names, in which case the file needs not to have any, so I need the `skip` again:

```r
read_delim(my_url, " ", col_names = c("instructor", "mark"), skip = 1)
```

```
Rows: 11 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): instructor
dbl (1): mark

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

# A tibble: 11 x 2
   instructor  mark
   <chr>      <dbl>
 1 ken           78
 2 ken           62
 3 ken           59
 4 ken           69
 5 ken           81
 6 thomas        83
 7 thomas        77
 8 thomas        63
 9 thomas        61
10 thomas        79
11 thomas        72
```

■

(b) * Obtain side-by-side boxplots of the scores for each class.

Solution

```r
library(tidyverse)
ggplot(marks, aes(x = class, y = score)) + geom_boxplot()
```

Remember: on a regular boxplot,[7] the groups go across (x), the variable measured goes up (y).

Extra: this might work:

```
ggplot(marks, aes(x = class, y = score)) + geom_boxplot() +
  coord_flip()
```

[7]Boxplots can also go across the page, but for us, they don't.

It does. That was a guess. So if you want sideways boxplots, this is how you can get them. Long group names sometimes fit better on the y-axis, in which case flipping the axes will help. (The `x` and `y` happen *before* the coordinate-flip, so they are the same as above, not the same way they come out.)

■

(c) Do the two classes appear to have similar or different scores, on average? Explain briefly.

Solution

The median for Thomas's class appears to be quite a bit higher than for Ken's class (the difference is actually about 6 marks). It's up to you whether you think this is a big difference or not: I want you to have *an* opinion, but I don't mind so much what that opinion is. Having said that the medians are quite a bit different, note that the boxes overlap substantially, so that the *distributions* of scores are pretty similar (or, the quartiles of scores are similar, or, the IQR of scores is similar for the two groups). If you say that, it's good, but I'm not insisting that you do.

■

(d) Obtain a boxplot of all the scores together, regardless of which class they came from.

Solution

Replace your x-coordinate by some kind of dummy thing like `1` (`factor(1)` also works):

```r
ggplot(marks, aes(x = 1, y = score)) + geom_boxplot()
```

The x-axis is kind of dopey, so you just ignore it. It is possible to remove it, but that is more work than it's worth, and I didn't get rid of the ticks below:

```r
ggplot(marks, aes(x = 1, y = score)) + geom_boxplot() +
  theme(
    axis.text.x = element_blank(),
    axis.title.x = element_blank()
  )
```

■

(e) Compute the median score (of all the scores together). Does this seem about right, looking at the boxplot? Explain briefly.

Solution

Three ways to get the median score. I like the first one best:

```
marks %>% summarize(med = median(score))
```

```
# A tibble: 1 x 1
    med
  <dbl>
1    72
```

```
with(marks, median(score))
```

```
[1] 72
```

```
median(marks$score)
```

```
[1] 72
```

`summarize` is the `tidyverse` "verb" that does what you want here. (The same idea gets the mean score for each class, below.)

The other ways use the basic function `median`. To make that work, you need to say that the variable `score` whose median you want lives in the data frame `marks`. These are two ways to do that.

Extra: if you wanted median by group, this is the approved (`tidyverse`) way:

```
marks %>%
  group_by(class) %>%
  summarize(med = median(score))
```

```
# A tibble: 2 x 2
  class    med
  <chr>  <dbl>
1 ken     69
2 thomas  74.5
```

To get something by group, the extra step is `group_by`, and then whatever you do after that is done for *each* group.

You can now go back and compare these medians with the ones on the boxplots in (here). They should be the same. Or you can even do this:

```
marks %>%
  group_by(class) %>%
  summarize(
    q1 = quantile(score, 0.25),
    med = median(score),
    q3 = quantile(score, 0.75)
  )
```

```
# A tibble: 2 x 4
  class     q1   med    q3
  <chr>  <dbl> <dbl> <dbl>
1 ken     62    69    78
2 thomas  65.2  74.5  78.5
```

You can calculate as many summaries as you like. These ones should match up with the top and bottom of the boxes on the boxplots. The only restriction is that the things on the right side of the equals should return a *single* number. If you have a function like `quantile` without anything extra that returns more than one number:

```
quantile(marks$score)
```

```
  0%  25%  50%  75% 100% 
59.0 62.5 72.0 78.5 83.0 
```

you're in trouble. Only read on if you *really* want to know how to handle this. Here's step 1:

```
marks %>%
  nest_by(class)
```

```
# A tibble: 2 x 2
# Rowwise:  class
  class                data
  <chr>  <list<tibble[,1]>>
1 ken               [5 x 1]
2 thomas            [6 x 1]
```

This is kind of a funky `group_by`. The things in the `data` column are the *whole* rest of the data frame: there were 5 students in Ken's class and 6 in Thomas's, and they each had a `score`, so 5 or 6 rows and 1 column. The column `data` is known in the trade as a "list-column".

Now, for each of those mini-data-frames, we want to calculate the quantiles of `score`. This is `rowwise`: for each of our mini-data-frames `data`, calculate the five-number summary of the column called `score` in *it*:

```
marks %>%
  nest_by(class) %>%
  rowwise() %>%
  mutate(qq = list(quantile(data$score)))
```

```
# A tibble: 2 x 3
# Rowwise:
  class                data qq
  <chr>  <list<tibble[,1]>> <list>
```

```
1 ken                [5 x 1] <dbl [5]>
2 thomas             [6 x 1] <dbl [5]>
```

I have to be a little bit careful about which data frame I want the `score` to come from: the ones hidden in `data`, which are the things we're for-eaching over.

This obtains a new list-column called `qq`, with the five-number summary for each group.[8]

Now we want to display the quantiles. This is the easiest way:

```
marks %>%
  nest_by(class) %>%
  rowwise() %>%
  mutate(qq = list(quantile(data$score))) %>%
  unnest(qq)
```

```
# A tibble: 10 x 3
   class                data    qq
   <chr>  <list<tibble[,1]>> <dbl>
 1 ken                [5 x 1]  59
 2 ken                [5 x 1]  62
 3 ken                [5 x 1]  69
 4 ken                [5 x 1]  78
 5 ken                [5 x 1]  81
 6 thomas             [6 x 1]  61
 7 thomas             [6 x 1]  65.2
 8 thomas             [6 x 1]  74.5
 9 thomas             [6 x 1]  78.5
10 thomas             [6 x 1]  83
```

The `unnest` turns the list-column back into actual data, so we get the five quantiles for each class.

The only thing this doesn't do is to show us which quantile is which (we know, of course, that the first one is the minimum, the last one is the max and the quartiles and median are in between). It would be nice to see which is which, though. A trick to do that is to use `enframe`, thus:

```
quantile(marks$score) %>% enframe()
```

[8] It's actually a coincidence that the five-number summary and Ken's class both have five values in them.

```
# A tibble: 5 x 2
  name  value
  <chr> <dbl>
1 0%     59
2 25%    62.5
3 50%    72
4 75%    78.5
5 100%   83
```

or thus:

```
enframe(quantile(marks$score))
```

```
# A tibble: 5 x 2
  name  value
  <chr> <dbl>
1 0%     59
2 25%    62.5
3 50%    72
4 75%    78.5
5 100%   83
```

I don't normally like the second way with all the brackets, but we'll be using it later.

The idea here is that the output from a quantile is a vector, but one with "names", namely the percentiles themselves. `enframe` makes a two-column data frame with the names and the values. (You can change the names of the columns it creates, but here we'll keep track of which is which.)

So we have a *two*-column data frame with a column saying which quantile is which. So let's rewrite our code to use this:

```
marks %>%
  nest_by(class) %>%
  rowwise() %>%
  mutate(qq = list(enframe(quantile(data$score))))
```

```
# A tibble: 2 x 3
# Rowwise:
  class                data qq
  <chr>  <list<tibble[,1]>> <list>
```

```
1 ken                [5 x 1] <tibble [5 x 2]>
2 thomas             [6 x 1] <tibble [5 x 2]>
```

Note that the `qq` data frames in the list-column now themselves have two columns.

And finally `unnest qq`:

```
marks %>%
  nest_by(class) %>%
  rowwise() %>%
  mutate(qq = list(enframe(quantile(data$score)))) %>%
  unnest(qq)
```

```
# A tibble: 10 x 4
   class                data name  value
   <chr>  <list<tibble[,1]>> <chr> <dbl>
 1 ken               [5 x 1] 0%     59
 2 ken               [5 x 1] 25%    62
 3 ken               [5 x 1] 50%    69
 4 ken               [5 x 1] 75%    78
 5 ken               [5 x 1] 100%   81
 6 thomas            [6 x 1] 0%     61
 7 thomas            [6 x 1] 25%    65.2
 8 thomas            [6 x 1] 50%    74.5
 9 thomas            [6 x 1] 75%    78.5
10 thomas            [6 x 1] 100%   83
```

Success! Or even:

```
marks %>%
  nest_by(class) %>%
  rowwise() %>%
  mutate(qq = list(enframe(quantile(data$score)))) %>%
  unnest(qq) %>%
  mutate(qn = parse_number(name)) %>%
  select(-name) %>%
  pivot_wider(names_from = qn, values_from = value)
```

```
# A tibble: 2 x 7
  class                data   `0`  `25`  `50`  `75` `100`
  <chr>  <list<tibble[,1]>> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
1 ken               [5 x 1]    59  62    69    78      81
2 thomas            [6 x 1]    61  65.2  74.5  78.5    83
```

This deliberately untidies the final answer to make it nicer to look at. (The lines before that create a numeric quantile, so that it sorts into the right order, and then get rid of the original quantile percents. Investigate what happens if you do a similar `pivot_wider` without doing that.)

3.13 Unprecedented rainfall

In 1997, a company in Davis, California, had problems with odour in its wastewater facility. According to a company official, the problems were caused by "unprecedented weather conditions" and "because rainfall was at 170 to 180 percent of its normal level, the water in the holding ponds took longer to exit for irrigation, giving it more time to develop an odour."

Annual rainfall data for the Davis area is here. Note that clicking on the link will display the file, and *right*-clicking on the link will give you some options, one of which is Copy Link Address, which you can then paste into your R Notebook.

The rainfall is measured in inches.

(a) Read in and display (some of) the data.

Solution

Look at the data file, and see that the values are separated by a single space, so `read_delim` will do it. Read straight from the URL; the hint above tells you how to copy it, which would even work if the link spans two lines.

```
my_url <- "http://ritsokiguess.site/datafiles/rainfall.txt"
rain <- read_delim(my_url, " ")
```

```
Rows: 47 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): Year, Rainfall

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
rain
```

```
# A tibble: 47 x 2
    Year Rainfall
   <dbl>    <dbl>
 1  1951     20.7
 2  1952     16.7
 3  1953     13.5
 4  1954     14.1
 5  1955     25.4
 6  1956     12.0
 7  1957     28.7
 8  1958     11.0
 9  1959     12.6
10  1960     12.8
# i 37 more rows
```

Note for later that the `Year` and the `Rainfall` have Capital Letters. You can call the data frame whatever you like, but I think something descriptive is better than eg. `mydata`.

Extra: this works because there is exactly one space between the year and the rainfall amount. But the year is always four digits, so the columns line up, and there is a space all the way down between the year and the rainfall. That means that this will also work:

```
my_url <- "http://ritsokiguess.site/datafiles/rainfall.txt"
rain <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  Year = col_double(),
  Rainfall = col_double()
)
```

```
rain
```

```
# A tibble: 47 x 2
    Year Rainfall
   <dbl>    <dbl>
 1  1951     20.7
 2  1952     16.7
 3  1953     13.5
 4  1954     14.1
```

```
 5  1955     25.4
 6  1956     12.0
 7  1957     28.7
 8  1958     11.0
 9  1959     12.6
10  1960     12.8
# i 37 more rows
```

This is therefore also good.

It also looks as if it could be tab-separated values, since the rainfall column always starts in the same place, but if you try it, you'll find that it doesn't work:

```
my_url <- "http://ritsokiguess.site/datafiles/rainfall.txt"
rain_nogood <- read_tsv(my_url)
```

```
Rows: 47 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): Year Rainfall

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
rain_nogood
```

```
# A tibble: 47 x 1
   `Year Rainfall`
   <chr>
 1 1951 20.66
 2 1952 16.72
 3 1953 13.51
 4 1954 14.1
 5 1955 25.37
 6 1956 12.05
 7 1957 28.74
 8 1958 10.98
 9 1959 12.55
10 1960 12.75
# i 37 more rows
```

This looks as if it worked, but it didn't, because there is only *one* column, of years and rainfalls smooshed together as text, and if you try to do anything else with them later it won't work.

Hence those values that might have been tabs actually were not. There's no way to be sure about this; you have to try something and see what works. An indication, though: if you have more than one space, and the things in the later columns are *left*-justified, that could be tab-separated; if the things in the later columns are *right*-justified, so that they finish in the same place but don't start in the same place, that is probably aligned columns.

■

(b) Summarize the data frame.

Solution

I almost gave the game away: this is `summary`.

```
summary(rain)
```

```
      Year         Rainfall
 Min.   :1951   Min.   : 6.14
 1st Qu.:1962   1st Qu.:12.30
 Median :1974   Median :16.72
 Mean   :1974   Mean   :18.69
 3rd Qu.:1986   3rd Qu.:25.21
 Max.   :1997   Max.   :37.42
```

The summary of the years may not be very helpful, but the summary of the annual rainfall values might be. It's not clear yet why I asked you to do this, but it will become clearer later.

■

(c) Make a suitable plot of the rainfall values. (We are not, for the time being, concerned about the years.)

Solution

This is one quantitative variable, so a histogram is your first thought. This means picking a number of bins. Not too many, since you want a picture of the shape:

```
ggplot(rain, aes(x=Rainfall)) + geom_histogram(bins=8)
```

If you picked fewer bins, you'll get a different picture:

```
ggplot(rain, aes(x=Rainfall)) + geom_histogram(bins=6)
```

The choice of the number of bins depends on what you think the story about shape is that you want to tell (see next part). You will probably need to try some different numbers of bins to see which one you like best. You can say something about what you tried, for example "I also tried 8 bins, but I like the histogram with 6 bins better."

■

(d) How would you describe the shape of the distribution of rainfall values?

Solution

This will depend on the histogram you drew in the previous part. If it looks like the first one, the best answer is "bimodal": that is, it has two peaks with a gap between them. If it looks like the second one, you have an easier time; this is ordinary right-skewness.

■

(e) In the quote at the beginning of the question, where do you think the assertion that the 1997 rainfall was "at 170 to 180 percent of its normal level" came from? Explain briefly.

Solution

First we need the 1997 rainfall. Go back and find it in the data. I am borrowing an idea from later in the course (because I am lazy):

```
rain %>% filter(Year==1997)
```

```
# A tibble: 1 x 2
   Year Rainfall
  <dbl>    <dbl>
1  1997     29.7
```

29.7 inches.

Now, what would be a "normal level" of rainfall? Some kind of average, like a mean or a median, maybe. But we have those, from our summary that we made earlier, repeated here for (my) convenience:

```
summary(rain)
```

```
      Year         Rainfall
 Min.   :1951   Min.   : 6.14
 1st Qu.:1962   1st Qu.:12.30
 Median :1974   Median :16.72
```

```
 Mean   :1974   Mean   :18.69
 3rd Qu.:1986   3rd Qu.:25.21
 Max.   :1997   Max.   :37.42
```

The mean is 18.69 and the median is 16.72 inches.

So divide the 1997 rainfall by each of the summaries, and see what happens, using your calculator, or using R as a calculator:

```
29.7/18.69
```

```
[1] 1.589085
```

```
29.7/16.72
```

```
[1] 1.776316
```

The 1997 rainfall was about 178 percent of the normal level if the normal level was the *median*.

■

(f) Do you think the official's calculation was reasonable? Explain briefly. (Note that this is not the same as asking whether the official's calculation was *correct*. This is an important distinction for you to make.)

Solution

There are several approaches to take. Argue for yours.

If you came to the conclusion that the distribution was right-skewed, you can say that the sensible "normal level" is the median, and therefore the official did the right thing. Using the mean would have been the wrong thing.

If you thought the distribution was bimodal, you can go a couple of ways: (i) it makes no sense to use any measure of location for "normal" (in fact, the mean rainfall is almost in that low-frequency bar, and so is not really a "normal level" at all). Or, (ii) it looks as if the years split into two kinds: low-rainfall years with around 15 inches, and high-rainfall years with more than 25 inches. Evidently 1997 was a high-rainfall year, but 29.7 inches was not especially high for a high-rainfall year, so the official's statement was an exaggeration. (I think (ii) is more insightful than (i), so ought to get more points.)

You could even also take a more conspiratorial approach and say that the official was trying to make 1997 look like a freak year, and picked the measure of location that made 1997 look more unusual.

"Normal level" here has nothing to do with a normal *distribution*; for this to make sense, the official would have needed to say something like "normal shape". This is why language skills are also important for a statistician to have.

■

(g) Do you think that the official was right to use the word "unprecedented" to describe the 1997 rainfall? Justify your answer briefly.

Solution

"Unprecedented" means "never seen before" or "never having happened or existed in the past".[9] That came out of my head; this link has a very similar "never before known or experienced").

If you look back at your histogram, there are several years that had over about 30 inches of rain: five or six, depending on your histogram. One of them was 1997, but there were others too, so 1997 was in no way "unprecedented".

Another approach that you have seen is to `View` your dataframe:

```
View(rain)
```

That will come out as a separate tab in your R Studio and you can look at it (yourself; it won't appear in the Preview). You can look at the 1997 rainfall (29.69 inches) and count how many were bigger than that, 4 of them. Or, save yourself some effort[10] and sort the rainfall values in descending order (with the biggest one first), by clicking on the little arrows next to Rainfall (twice). Mine looks like this:

[9] Searching for "define" followed by a word is a good way to find out exactly what that word means, if you are not sure, but you should at least say where you got the definition from if you had to look it up.

[10] When you have a computer at your disposal, it's worth taking a few minutes to figure out how to use it to make your life easier.

	Year	Rainfall
47	1982	37.42
46	1981	31.29
45	1994	31.22
44	1992	29.84
43	1997	29.69
42	1957	28.74
40	1972	27.69
41	1977	27.69
39	1966	27.64
38	1985	27.47
37	1962	27.10
36	1955	25.37
35	1979	25.06
34	1968	24.67
33	1995	24.50
32	1973	20.87
31	1951	20.66
30	1996	19.52
29	1964	18.56
28	1991	17.46

Later, we learn how to sort in code, which goes like this (to sort highest first):

```
rain %>% arrange(desc(Rainfall))
```

```
# A tibble: 47 x 2
    Year Rainfall
   <dbl>    <dbl>
 1  1982     37.4
 2  1981     31.3
 3  1994     31.2
 4  1992     29.8
 5  1997     29.7
 6  1957     28.7
 7  1972     27.7
 8  1977     27.7
 9  1966     27.6
10  1985     27.5
# i 37 more rows
```

A more sophisticated way that we learn later:

```r
rain %>% summarize(max=max(Rainfall))
```

```
# A tibble: 1 x 1
    max
  <dbl>
1  37.4
```

This is greater than the rainfall for 1997, ruling out "unprecedented".

1997 was only the *fifth* highest rainfall, and two of the higher ones were also in the 1990s. Definitely not "unprecedented". The official needs to get a new dictionary!

■

3.14 Learning algebra

At a high school in New Jersey, teachers were interested in what might help students to learn algebra. One idea was laptops as a learning aid, to see whether having access to one helped with algebra scores. (This was some time ago.) The 20 students in one class were given laptops to use in school and at home, while the 27 students in another class were not given laptops. For all of these students, the final exam score in algebra was recorded. The data are in http://ritsokiguess.site/datafiles/algebra.txt, with two columns, one indicating whether the student received a laptop or not, and the other giving their score on the algebra final exam.

(a) Read in and display (some of) the data. Do you have (i) the correct number of observations, and (ii) the correct *type* of columns? Explain briefly.

Solution

Take a look at the data file first: the data values are *aligned in columns* with variable numbers of spaces between, so `read_table` is the thing. Read directly from the URL, rather than trying to copy the data from the website:

```r
my_url <- "http://ritsokiguess.site/datafiles/algebra.txt"
algebra <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  laptop = col_character(),
  score = col_double()
)
```

```r
algebra
```

```
# A tibble: 47 x 2
   laptop score
   <chr>  <dbl>
 1 yes       98
 2 yes       84
 3 yes       97
 4 yes       93
 5 yes       88
 6 yes       57
 7 yes      100
 8 yes       84
 9 yes      100
10 yes       81
# i 37 more rows
```

There were $20 + 27 = 47$ students altogether in the two classes, and we do indeed have 47 rows, one per student. So we have the right number of rows. This is two independent samples; each student was in only one of the two classes, either the class whose students got laptops or not. The values in the `laptop` column are text (see the `chr` at the top), and the values in the `score` column are numbers (`dbl` at the top). Alternatively, you can look at the R Console output in which you see that `laptop` is `col_character()` (text) and `score` is `col_double()` (numerical, strictly a decimal number).

Extra 1: `read.table` also works but it is *wrong* in this course (because it is not what I taught you in class).

Extra 2: with more than one space between the values, `read_delim` will not work. Or, perhaps more confusing, it will appear to work and then fail later, which means that you need to pay attention:

```r
d <- read_delim(my_url, " ")
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for details,
e.g.:
  dat <- vroom(...)
  problems(dat)
```

```
Rows: 47 Columns: 2
-- Column specification --------------------------------------------------------
```

```
Delimiter: " "
chr (2): laptop, score

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
d
```

```
# A tibble: 47 x 2
   laptop score
   <chr>  <chr>
 1 yes    "    98"
 2 yes    "    84"
 3 yes    "    97"
 4 yes    "    93"
 5 yes    "    88"
 6 yes    "    57"
 7 yes    "   100"
 8 yes    "    84"
 9 yes    "   100"
10 yes    "    81"
# i 37 more rows
```

This *looks* all right, but look carefully: the `laptop` column is correctly text, but the `score` column, which should be numbers (`dbl`), is actually text as well. An easier way to see this is to look at the output from the console, which is the descriptions of the columns: they are *both* `col_character` or text, while `score` should be numbers. You might be able to see exactly what went wrong: with more than one space separating the values, the remaining spaces went into `score`, which then becomes a piece of text with some spaces at the front and then numbers.

This will actually work for a while, as you go through the question, but will come back to bite you the moment you need `score` to be numerical (eg. when you try to draw a boxplot), because it is actually not numerical at all.

Extra 3: this is the standard R way to lay out this kind of data, with all the outcome values in one column and a second (categorical) column saying which group each observation was in. In other places you might see two separate columns of scores, one for the students with laptops and one for the students without, as below (you won't understand the code below now, but you will by the end of the course):

```r
algebra %>%
mutate(row = c(1:20, 1:27)) %>%
```

```r
pivot_wider(names_from = laptop, values_from = score)
```

```
# A tibble: 27 x 3
     row   yes    no
   <int> <dbl> <dbl>
 1     1    98    63
 2     2    84    83
 3     3    97    97
 4     4    93    93
 5     5    88    52
 6     6    57    74
 7     7   100    83
 8     8    84    63
 9     9   100    88
10    10    81    86
# i 17 more rows
```

A column of `yes` and a column of `no`. The classes were of different sizes, so the `yes` column, with only 20 observations, has some `NA` ("missing") observations at the end (scroll down to see them) to enable the dataframe to keep a rectangular shape.

We will learn later what to call these layouts of data: "longer" and "wider" (respectively), and how to convert between them. R usually likes "longer" data, as in the data file, but you will often see data sets displayed wider because it takes up less space.

■

(b) Make a suitable graph of these data.

Solution

The teachers were hoping to see how the laptop-yes and the laptop-no groups compared in terms of algebra scores, so side-by-side boxplots would be helpful. More simply, we have one quantitative and one categorical variable, which is a boxplot according to the table in the notes:

```r
ggplot(algebra, aes(x = laptop, y = score)) + geom_boxplot()
```

Extra: as you will note below, the median score for the students with laptops is a little higher for the students who had laptops. This is easy to see on a boxplot because that is what a boxplot does. (That was what Tukey, who we will meet later, *designed* the boxplot to do.)

Another plot you might have drawn is a histogram for each group, side by side, or, as they come out here, above and below. This works using facets:

```
ggplot(algebra, aes(x = score)) +
geom_histogram(bins = 10) +
facet_wrap(~laptop, ncol = 1)
```

Looking at those, can you *really* say that the median is slightly higher for the `yes` group? I really don't think you can. Certainly it is clear from the histograms that the spread for the `yes` group is less, but comparing the medians is much more easily done from the boxplot. The teachers were interested in whether the laptops were associated with higher scores on average, so the kind of comparison that the boxplot affords is clearly preferred here.

If you are interested in the code: you imagine you're going to make a histogram of scores regardless of group, and then at the end you facet by your grouping variable. I added the `ncol = 1` to make the plots come out in one column (that is, one above the other). If you don't do this, they come out left and right, which makes the distributions even harder to compare.

■

(c) Comment briefly on your graph, thinking about what the teachers would like to know.

Solution

There are three things to say something about, the first two of which would probably interest the teachers:

- comparison of centre: the *median* score for the group that had laptops is (slightly) higher than for the group that did not.
- comparison of spread: the scores for the group that had laptops are less spread out (have smaller *interquartile range*) than for the group that did not.
- assessment of shape: both groups have low outliers, or are skewed to the left in shape.

Some comments from me:

- boxplots say *nothing* about mean and standard deviation, so don't mention those here. You should say something about the measures of centre (median) and spread (IQR) that they *do* use.
- I think of skewness as a property of a whole distribution, but outlierness as a property of individual observations. So, when you're looking at this one, think about where the evidence about shape is coming from: is it coming from those one or two low values that are different from the rest (which would be outliers), or is it coming from the whole distribution (would you get the same story if those maybe-outliers are taken away)? My take is that if you take the outliers away, both distributions are close to symmetric, and therefore what you see here is outliers rather than skewness. If you see something different, make the case for it.

One reason to suspect skewness or something like it is that test scores have an upper limit (100) that some of the scores got close to, and no effective lower limit (the lower limit is 0 but no-one got very close to that). In this sort of situation, you'd expect the scores to be skewed away from the limit: that is, to the left. Or to have low outliers rather than high ones.

■

(d) Work out the median and inter-quartile range for the students who did and who did not have laptops, and compare with the boxplot. (In R, the inter-quartile range is `IQR` in uppercase.)

Solution

This is easy to make way harder than it needs to be: `group_by` and `summarize` will do it. Put the two summaries in one `summarize`:

```
algebra %>%
group_by(laptop) %>%
summarize(med = median(score), iqr = IQR(score))
```

```
# A tibble: 2 x 3
  laptop   med   iqr
  <chr>  <dbl> <dbl>
1 no        81    20
2 yes       84    13
```

Then relate these to the information on the boxplot: the centre line of the box is the median. For the `no` group this is just above 80, so 81 makes sense; for the `yes` group this is not quite halfway between 80 and 90, so 84 makes sense.

The inter-quartile range is the height of the box for each group. Estimate the top and bottom of the two boxes from the boxplot scale, and subtract. For the `no` group this is something like $88 - 68$ which *is* 20, and for the `yes` group it is something like $93 - 80$ which is indeed 13.

Extra: I didn't ask you here about whether the difference was likely meaningful. The focus here was on getting the graph and summaries. If I had done so, you would then need to consider things like whether a three-point difference in medians could have been chance, and whether we really had random allocation of students to groups.

To take the second point first: these are students who chose to take two different classes, rather than being randomly allocated to classes as would be the case in a true experiment. What we have is really in between an experiment and an observational study; yes, there was a treatment (laptop or not) that was (we hope) randomly allocated to one class and not the other, but the classes could have been different for any number of other reasons that had nothing to do with having laptops or not, such as time of day, teacher, approach to material, previous ability at algebra, etc.

So even if we are willing to believe that the students were as-if randomized to laptop or not, the question remains as to whether that three-point difference in medians is reproducible or indicative of a real difference or not. This is the kind of thing we would try a two-sample t-test with. In this case, we might doubt whether it will come out significant (because of the small difference in medians and presumably means, compared to the amount of variability present), and, even then, there is the question of whether we *should* be doing a t-test at all, given the outliers.

■

4 One-sample inference

```
library(tidyverse)
```

4.1 Hunter-gatherers in Australia

A hunter-gatherer society is one where people get their food by hunting, fishing or foraging rather than by agriculture or by raising animals. Such societies tend to move from place to place. Anthropologists have studied hunter-gatherer societies in forest ecosystems across the world. The average population density of these societies is 7.38 people per 100 km^2. Hunter-gatherer societies on different continents might have different population densities, possibly because of large-scale ecological constraints (such as resource availability), or because of other factors, possibly social and/or historic, determining population density.

Some hunter-gatherer societies in Australia were studied, and the population density per 100 km^2 recorded for each. The data are in http://ritsokiguess.site/datafiles/hg.txt.

(a) Read the data into R. Do you have the correct variables? How many hunter-gatherer societies in Australia were studied? Explain briefly.

(b) The question of interest is whether these Australian hunter-gatherer societies are like the rest of the world in terms of mean population density. State suitable null and alternative hypotheses. *Define any symbols you use*: that is, if you use a symbol, you also have to say what it means.

(c) Test your hypotheses using a suitable test. What do you conclude, in the context of the data?

(d) Do you have any doubts about the validity of your test? Explain briefly, using a suitable graph to support your explanation.

4.2 Buses to Boulder

A bus line operates a route from Denver to Boulder (these places are in Colorado). The schedule says that the journey time should be 60 minutes. 11 randomly chosen journey times were recorded, and these are in the file link, with journey times shown in minutes.

(a) Read the data into R, and display the data frame that you read in.

(b) Run a suitable test to see whether there is evidence that the mean journey time differs from 60 minutes. What do you conclude? (I want a conclusion that says something about journey times of buses.)

(c) Give a 95% confidence interval for the mean journey time. (No R code is needed here.)

(d) Do you draw consistent conclusions from your test and confidence interval? Explain briefly.

(e) Draw a boxplot of the journey times. Do you see a reason to doubt the test that you did above?

4.3 Length of gestation in North Carolina

The data in file link are about 500 randomly chosen births of babies in North Carolina. There is a lot of information: not just the weight at birth of the baby, but whether the baby was born prematurely, the ages of the parents, whether the parents are married, how long (in weeks) the pregnancy lasted (this is called the "gestation") and so on. We have seen these data before.

(a) Read in the data from the file into R, bearing in mind what type of file it is.

(b) Find a 95% confidence interval for the mean birth weight of all babies born in North Carolina (of which these babies are a sample). At the end, you should state what the confidence interval is. Giving some output is necessary, but *not* enough by itself.

(c) Birth weights of babies born in the United States have a mean of 7.3 pounds. Is there any evidence that babies born in North Carolina are less heavy on average? State appropriate hypotheses, do your test, obtain a P-value and state your conclusion, in terms of the original data.

(d) The theory behind the t-test says that the distribution of birth weights should be (approximately) normally distributed. Obtain a histogram of the birth weights. Does it look approximately normal? Comment briefly. (You'll have to pick a number of bins for your histogram first. I don't mind very much what you pick, as long as it's not obviously too many or too few bins.)

4.4 Inferring ice break-up in Nenana

Nenana, Alaska, is about 50 miles west of Fairbanks. Every spring, there is a contest in Nenana. A wooden tripod is placed on the frozen river, and people try to guess the exact minute when the ice melts enough for the tripod to fall through the ice. The contest started in 1917 as an amusement for railway workers, and has taken place every year since. Now, hundreds of thousands of people enter their guesses on the Internet and the prize for the winner can be as much as $300,000.

Because so much money is at stake, and because the exact same tripod is placed at the exact same spot on the ice every year, the data are consistent and accurate. The data are in link.

Yes, we saw these data before.

(a) Read the data into R, as before, or use the data frame that you read in before. Note that the values are separated by *tabs* rather than spaces, so you'll need an appropriate `read_` to read it in.

(b) Obtain a 90% confidence interval for the mean `JulianDate`. What interval do you get? Looking back at your histogram, do you have any doubts about the validity of what you have just done?

(c) An old-timer in Nenana strokes his grey beard and says "When I were young, I remember the tripod used to fall into the water around May 10". In a non-leap year, May 10 is Julian day 130. Test the null hypothesis that the mean `JulianDay` is 130, against the alternative that it is less. What do you conclude? What practical implication does that have (assuming that the old-timer has a good memory)?

(d) Plot `JulianDate` against `Year` on a scatterplot. What recent trends, if any, do you see? Comment briefly. (You did this before, but I have some extra comments on the graph this time, so feel free to just read this part.)

4.5 Diameters of trees

The Wade Tract in Thomas County, Georgia, is an old-growth forest of longleaf pine trees. It has survived in a relatively undisturbed state since before settlements of the area by Europeans. For each tree in the tract, researchers measured the diameter at breast height. This is a standard measure in forestry: it is defined as the diameter of the tree at 4.5 feet above the ground.[1] They are interested in the mean diameter at breast height of the trees in this tract. These values are in http://ritsokiguess.site/datafiles/treediameter.csv. The diameters are measured in centimetres. The easiest way to get the URL is to *right*-click on the blue text

[1]The height of a typical human breast off the ground. Men have a breast too, you know.

and select Copy URL. (If you copy and paste the actual text you might end up with extra spaces, especially if the printed URL goes over two lines.)

(a) Read in and display (some of) the data.

(b) Make a suitable plot of your dataframe.

(c) Obtain a 95% confidence interval for the mean diameter.

(d) Based on what you have seen so far, would you expect to reject a null hypothesis that the population mean diameter (of all longleaf pines like these) is 35 cm? Explain briefly. Then, carry out the test (against a two-sided alternative) and explain briefly whether you were right.

(e) Would you expect 35 cm to be in a *99%* confidence interval for the mean diameter? Explain briefly, and then see if you were right.

4.6 One-sample cholesterol

The data set here contains cholesterol measurements for heart attack patients (at several different times) as well as for a group of control patients. We will focus on the control patients in this question.

(a) Read in and display (some of) the data.

(b) Make a suitable plot of the cholesterol levels of the control patients, and comment briefly on the shape of the distribution.

(c) It is recommended that people in good health, such as the Control patients here, keep their cholesterol level below 200. Is there evidence that the mean cholesterol level of the population of people of which the Control patients are a sample is less than 200? Show that you understand the process, and state your conclusion in the context of the data.

(d) What values could the population mean cholesterol level take? You might need to get some more output to determine this.

(e) Explain briefly why you would be reasonably happy to trust the t procedures in this question. (There are two points you need to make.)

My solutions follow:

4.7 Hunter-gatherers in Australia

A hunter-gatherer society is one where people get their food by hunting, fishing or foraging rather than by agriculture or by raising animals. Such societies tend to move from place to place. Anthropologists have studied hunter-gatherer societies in forest ecosystems across the world. The average population density of these societies is 7.38 people per 100 km^2. Hunter-gatherer societies on different continents might have different population densities, possibly because of large-scale ecological constraints (such as resource availability), or because of other factors, possibly social and/or historic, determining population density.

Some hunter-gatherer societies in Australia were studied, and the population density per 100 km^2 recorded for each. The data are in http://ritsokiguess.site/datafiles/hg.txt.

(a) Read the data into R. Do you have the correct variables? How many hunter-gatherer societies in Australia were studied? Explain briefly.

Solution

The data values are separated by (single) spaces, so `read_delim` is the thing:

```
url="http://ritsokiguess.site/datafiles/hg.txt"
societies=read_delim(url," ")
```

```
Rows: 13 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): name
dbl (1): density

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

I like to put the URL in a variable first, because if I don't, the `read_delim` line can be rather long. But if you want to do it in one step, that's fine, as long as it's clear that you are doing the right thing.

Let's look at the data frame:

```
societies
```

```
# A tibble: 13 x 2
   name       density
   <chr>        <dbl>
```

```
 1 jeidji       17
 2 kuku         50
 3 mamu         45
 4 ngatjan      59.8
 5 undanbi      21.7
 6 jinibarra    16
 7 ualaria       9
 8 barkindji    15.4
 9 wongaibon     5.12
10 jaralde      40
11 tjapwurong   35
12 tasmanians   13.4
13 badjalang    13.4
```

I have the name of each society and its population density, as promised (so that is correct). There were 13 societies that were studied. For me, they were all displayed. For you, you'll probably see only the first ten, and you'll have to click Next to see the last three.

■

(b) The question of interest is whether these Australian hunter-gatherer societies are like the rest of the world in terms of mean population density. State suitable null and alternative hypotheses. *Define any symbols you use*: that is, if you use a symbol, you also have to say what it means.

Solution

The mean for the world as a whole ("average", as stated earlier) is 7.38. Let μ denote the population mean for Australia (of which these societies are a sample). Then our hypotheses are:

$$H_0 : \mu = 7.38$$

and

$$H_a : \mu \neq 7.38.$$

There is no reason for a one-sided alternative here, since all we are interested in is whether Australia is different from the rest of the world. *Expect to lose a point* if you use the symbol μ without saying what it means.

■

(c) Test your hypotheses using a suitable test. What do you conclude, in the context of the data?

Solution

A t-test, since we are testing a mean:

```r
t.test(societies$density,mu=7.38)
```

```
    One Sample t-test

data:  societies$density
t = 3.8627, df = 12, p-value = 0.002257
alternative hypothesis: true mean is not equal to 7.38
95 percent confidence interval:
 15.59244 36.84449
sample estimates:
mean of x
 26.21846
```

The P-value is 0.0023, less than the usual α of 0.05, so we *reject* the null hypothesis and conclude that the mean population density is not equal to 7.38. That is to say, Australia is different from the rest of the world in this sense.

As you know, "reject the null hypothesis" is only part of the answer, so gets only part of the marks.

■

(d) Do you have any doubts about the validity of your test? Explain briefly, using a suitable graph to support your explanation.

Solution

The assumption behind the t-test is that the data are approximately normal. We can assess that in several ways, but the simplest (which is perfectly acceptable at this point) is a histogram. You'll need to pick a suitable number of bins. This one comes from Sturges' rule:

```r
ggplot(societies,aes(x=density))+geom_histogram(bins=5)
```

Your conclusion might depend on how many bins you chose for your histogram. Here's 8 bins (which is really too many with only 13 observations, but it actually shows the shape well):

```
ggplot(societies,aes(x=density))+geom_histogram(bins=8)
```

or you can get a number of bins from one of the built-in functions, such as:

```
mybins=nclass.FD(societies$density)
mybins
```

```
[1] 3
```

This one is small. The interquartile range is large and n is small, so the binwidth will be large and therefore the number of bins will be small.

Other choices: a one-group boxplot:

```
ggplot(societies,aes(x=1,y=density))+geom_boxplot()
```

This isn't the best for assessing normality as such, but it will tell you about lack of symmetry and outliers, which are the most important threats to the t-test, so it's fine here. Or, a normal quantile plot:

```
ggplot(societies,aes(sample=density))+
stat_qq()+stat_qq_line()
```

This is actually the best way to assess normality, but I'm not expecting you to use this plot here, because we may not have gotten to it in class yet. (If you have read ahead and successfully use the plot, it's fine.)

After you have drawn your chosen plot (you need *one* plot), you need to say something about normality and thus whether you have any doubts about the validity of your t-test. This will depend on the graph you drew: if you think your graph is symmetric and outlier-free, you should have no doubts about your t-test; if you think it has something wrong with it, you should say what it is and express your doubts. My guess is that you will think this distribution is skewed to the right. Most of my plots are saying that.[2]

On the website where I got these data, they were using the data as an example for another test, precisely *because* they thought the distribution was right-skewed. Later on, we'll learn about the sign test for the median, which I think is actually a better test here.

■

4.8 Buses to Boulder

A bus line operates a route from Denver to Boulder (these places are in Colorado). The schedule says that the journey time should be 60 minutes. 11 randomly chosen journey times

[2]The normal quantile plot is rather interesting: it says that the uppermost values are approximately normal, but the *smallest* eight or so values are too bunched up to be normal. That is, normality fails not because of the long tail on the right, but the bunching on the left. Still right-skewed, though.

were recorded, and these are in the file link, with journey times shown in minutes.

(a) Read the data into R, and display the data frame that you read in.

Solution

Since you can read the data directly from the URL, do that (if you are online) rather than having to copy and paste and save, and then find the file you saved. Also, there is only one column, so you can pretend that there were multiple columns, separated by whatever you like. It's least typing to pretend that they were separated by commas like a `.csv` file:

```
my_url <- "http://ritsokiguess.site/datafiles/buses.txt"
journey.times <- read_csv(my_url)
```

```
Rows: 11 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (1): minutes

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
journey.times
```

```
# A tibble: 11 x 1
   minutes
     <dbl>
 1      58
 2      61
 3      69
 4      62
 5      81
 6      54
 7      72
 8      71
 9      53
10      54
11      66
```

Using `read_delim` with any delimiter (such as `" "`) will also work, and is thus also good.

Variable names in R can have a dot (or an underscore, but not a space) in them. I have grown accustomed to using dots to separate words. This works in R but not other languages, but is seen by some as old-fashioned, with underscores being the modern way.[3] You can also use what is called "camel case" by starting each "word" after the first with an uppercase letter like this:

```
journeyTimes <- read_csv(my_url)
```

You have to get the capitalization and punctuation right when you use your variables, no matter what they're called. In any of the cases above, there is no variable called `journeytimes`. As Jenny Bryan (in link) puts it, boldface in original: Implicit contract with the computer / scripting language: Computer will do tedious computation for you. In return, you will be completely precise in your instructions. Typos matter. Case matters. **Get better at typing.**

■

(b) Run a suitable test to see whether there is evidence that the mean journey time differs from 60 minutes. What do you conclude? (I want a conclusion that says something about journey times of buses.)

Solution

`t.test` doesn't take a `data=` to say which data frame to use. Wrap it in a `with`:

```
with(journey.times, t.test(minutes, mu = 60))
```

```
    One Sample t-test

data:  minutes
t = 1.382, df = 10, p-value = 0.1971
alternative hypothesis: true mean is not equal to 60
95 percent confidence interval:
 57.71775 69.73680
sample estimates:
mean of x
 63.72727
```

We are testing that the mean journey time is 60 minutes, against the two-sided alternative (default) that the mean is not equal to 60 minutes. The P-value, 0.1971, is a lot bigger than

[3] In some languages, a dot is used to concatenate bits of text, or as a way of calling a method on an object. But in R, a dot has no special meaning, and is used in function names like `t.test`. Or `p.value`.

the usual α of 0.05, so we cannot reject the null hypothesis. That is, there is no evidence that the mean journey time differs from 60 minutes.

As you remember, we have not proved that the mean journey time *is* 60 minutes, which is what "accepting the null hypothesis" would be. We have only failed to reject it, in a shoulder-shrugging kind of way: "the mean journey time *could* be 60 minutes". The other acceptable word is "retain"; when you say "we retain the null hypothesis", you imply something like "we act as if the mean is 60 minutes, at least until we find something better."

■

(c) Give a 95% confidence interval for the mean journey time. (No R code is needed here.)

Solution

Just read it off from the output: 57.72 to 69.74 minutes.

■

(d) Do you draw consistent conclusions from your test and confidence interval? Explain briefly.

Solution

The test said that we should not reject a mean of 60 minutes. The confidence interval says that 60 minutes is inside the interval of plausible values for the population mean, which is another way of saying the same thing. (If we had rejected 60 as a mean, 60 would have been *outside* the confidence interval.)

■

(e) Draw a boxplot of the journey times. Do you see a reason to doubt the test that you did above?

Solution

The grouping variable is a "nothing" as in the Ken and Thomas question (part (d)):

```
ggplot(journey.times, aes(x = 1, y = minutes)) + geom_boxplot()
```

The assumption behind the t-test is that the population from which the data come has a normal distribution: ie. symmetric with no outliers. A small sample (here we have 11 values) even from a normal distribution might look quite non-normal (as in Assignment 0 from last week), so I am not hugely concerned by this boxplot. However, it's perfectly all right to say that this distribution is skewed, and therefore we should doubt the t-test, because the upper whisker is longer than the lower one. In fact, the topmost value is very nearly an outlier:[4]

```
ggplot(journey.times, aes(x = minutes)) + geom_histogram(bins = 5)
```

[4]Whether you think it is or not may depend on how many bins you have on your histogram. With 5 bins it looks like an outlier, but with 6 it does not. Try it and see.

and there might be skewness as well, so maybe I should have been concerned.

I would be looking for some intelligent comment on the boxplot: what it looks like vs. what it ought to look like. I don't so much mind what that comment is, as long as it's intelligent enough.

Perhaps I should draw a normal quantile plot:

```
ggplot(journey.times, aes(sample = minutes)) + stat_qq() + stat_qq_line()
```

The normal quantile plot is saying that the problem is actually at the *bottom* of the distribution: the lowest value is not low enough, but the highest value is actually *not* too high. So this one seems to be on the edge between OK and being right-skewed (too bunched up at the bottom). My take is that with this small sample this is not too bad. But you are free to disagree.

If you don't like the normality, you'd use a *sign test* and test that the *median* is not 60 minutes, which you would (at my guess) utterly fail to reject:

```
library(smmr)
sign_test(journey.times, minutes, 60)
```

```
$above_below
below above
    4     7

$p_values
  alternative   p_value
1       lower 0.8867187
2       upper 0.2744141
3   two-sided 0.5488281
```

```
ci_median(journey.times, minutes)
```

```
[1] 54.00195 71.99023
```

and so we do. The median could easily be 60 minutes.

■

4.9 Length of gestation in North Carolina

The data in file link are about 500 randomly chosen births of babies in North Carolina. There is a lot of information: not just the weight at birth of the baby, but whether the baby was born prematurely, the ages of the parents, whether the parents are married, how long (in weeks) the pregnancy lasted (this is called the "gestation") and so on. We have seen these data before.

(a) Read in the data from the file into R, bearing in mind what type of file it is.

Solution

This is a `.csv` file (it came from a spreadsheet), so it needs reading in accordingly. Work directly from the URL (rather than downloading the file):

```r
myurl <- "http://ritsokiguess.site/datafiles/ncbirths2.csv"
bw <- read_csv(myurl)
```

```
Rows: 500 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (10): father_age, mother_age, weeks_gestation, pre_natal_visits, marital...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

■

(b) Find a 95% confidence interval for the mean birth weight of all babies born in North Carolina (of which these babies are a sample). At the end, you should state what the confidence interval is. Giving some output is necessary, but *not* enough by itself.

Solution

This:

```r
t.test(bw$weight_pounds)
```

```
    One Sample t-test

data:  bw$weight_pounds
t = 104.94, df = 499, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 6.936407 7.201093
sample estimates:
mean of x
  7.06875
```

or (the same, but remember to match your brackets):

```
with(bw, t.test(weight_pounds))
```

```
    One Sample t-test

data:  weight_pounds
t = 104.94, df = 499, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 6.936407 7.201093
sample estimates:
mean of x
  7.06875
```

The confidence interval goes from 6.94 to 7.20 pounds.

There is an annoyance about `t.test`. Sometimes you can use `data=` with it, and sometimes not. When we do a two-sample t-test later, there is a "model formula" with a squiggle in it, and there we can use `data=`, but here not, so you have to use the dollar sign or the `with` to say which data frame to get things from. The distinction seems to be that *if you are using a model formula*, you can use `data=`, and if not, not.

This is one of those things that is a consequence of R's history. The original `t.test` was without the model formula and thus without the `data=`, but the model formula got "retro-fitted" to it later. Since the model formula comes from things like regression, where `data=` is legit, that had to be retro-fitted as well. Or, at least, that's my understanding.

■

(c) Birth weights of babies born in the United States have a mean of 7.3 pounds. Is there any evidence that babies born in North Carolina are less heavy on average? State appropriate hypotheses, do your test, obtain a P-value and state your conclusion, in terms of the original data.

Solution

Let μ be the population mean (the mean weight of all babies born in North Carolina). Null hypothesis is $H_0 : \mu = 7.3$ pounds, and the alternative is that the mean is less: $H_a : \mu < 7.3$ pounds.

Note that I defined μ first before I used it.

This is a one-sided alternative, which we need to feed into `t.test`:

```
t.test(bw$weight_pounds, mu = 7.3, alternative = "less")
```

```
    One Sample t-test

data:  bw$weight_pounds
t = -3.4331, df = 499, p-value = 0.0003232
alternative hypothesis: true mean is less than 7.3
95 percent confidence interval:
     -Inf 7.179752
sample estimates:
mean of x
  7.06875
```

Or with `with`. If you see what I mean.

The P-value is 0.0003, which is *less* than any α we might have chosen: we *reject* the null hypothesis in favour of the alternative, and thus we conclude that the mean birth weight of babies in North Carolina is indeed less than 7.3 pounds.

"Reject the null hypothesis" is *not* a complete answer. You need to say something about what rejecting the null hypothesis means *in this case*: that is, you must make a statement about birth weights of babies.

■

(d) The theory behind the t-test says that the distribution of birth weights should be (approximately) normally distributed. Obtain a histogram of the birth weights. Does it look approximately normal? Comment briefly. (You'll have to pick a number of bins for your histogram first. I don't mind very much what you pick, as long as it's not obviously too many or too few bins.)

Solution

We did this before (and discussed the number of bins before), so I'll just reproduce my 10-bin histogram (which is what I preferred, but this is a matter of taste):

```
ggplot(bw, aes(x = weight_pounds)) + geom_histogram(bins = 10)
```

So, we were assessing normality. What about that?

It is mostly normal-looking, but I am suspicious about those *very* low birth weights, the ones below about 4 pounds. There are too many of those, as I see it.

If you think this is approximately normal, you need to make some comment along the lines of "the shape is approximately symmetric with no outliers". I think my first answer is better, but this answer is worth something, since it is a not completely unreasonable interpretation of the histogram.

A normal quantile plot is better for assessing normality than a histogram is, but I won't make you do one until we have seen the idea in class. Here's the normal quantile plot for these data:

```
ggplot(bw, aes(sample = weight_pounds)) + stat_qq() + stat_qq_line()
```

This is rather striking: the lowest birthweights (the ones below 5 pounds or so) are *way* too low for a normal distribution to apply. The top end is fine (except perhaps for that one very heavy baby), but there are too many low birthweights for a normal distribution to be believable. Note how much clearer this story is than on the histogram.

Having said that, the t-test, especially with a sample size as big as this (500), behaves *very* well when the data are somewhat non-normal (because it takes advantage of the Central Limit Theorem: that is, it's the *sampling distribution of the sample mean* whose shape matters). So, even though the data are definitely not normal, I wouldn't be too worried about our test.

This perhaps gives some insight as to why Freedman-Diaconis said we should use so many bins for our histogram. We have a lot of low-end outliers, so that the IQR is actually *small* compared to the overall spread of the data (as measured, say, by the SD or the range) and so FD thinks we need a lot of bins to describe the shape. Sturges is based on data being approximately normal, so it will tend to produce a small number of bins for data that have outliers.

■

4.10 Inferring ice break-up in Nenana

Nenana, Alaska, is about 50 miles west of Fairbanks. Every spring, there is a contest in Nenana. A wooden tripod is placed on the frozen river, and people try to guess the exact minute when

the ice melts enough for the tripod to fall through the ice. The contest started in 1917 as an amusement for railway workers, and has taken place every year since. Now, hundreds of thousands of people enter their guesses on the Internet and the prize for the winner can be as much as $300,000.

Because so much money is at stake, and because the exact same tripod is placed at the exact same spot on the ice every year, the data are consistent and accurate. The data are in link.

Yes, we saw these data before.

(a) Read the data into R, as before, or use the data frame that you read in before. Note that the values are separated by *tabs* rather than spaces, so you'll need an appropriate `read_` to read it in.

Solution

These are "tab-separated values", so `read_tsv` is the thing, as for the Australian athletes:

```
myurl <- "http://ritsokiguess.site/datafiles/nenana.txt"
nenana <- read_tsv(myurl)
```

```
Rows: 87 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): Date&Time
dbl (2): Year, JulianDate

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Use whatever name you like for the data frame. One that is different from any of the column headers is smart; then it is clear whether you mean the whole data frame or one of its columns. `ice` or `melt` or anything like that would also be good.

■

(b) Obtain a 90% confidence interval for the mean `JulianDate`. What interval do you get? Looking back at your histogram, do you have any doubts about the validity of what you have just done?

Solution

This is a matter of using `t.test` and pulling out the interval. Since we are looking for a non-standard interval, we have to remember `conf.level` as the way to get the confidence

level that we want. I'm going with `with` this time, though the dollar-sign thing is equally as good:

```
with(nenana, t.test(JulianDate, conf.level = 0.90))
```

```
    One Sample t-test

data:  JulianDate
t = 197.41, df = 86, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 124.4869 126.6018
sample estimates:
mean of x
 125.5443
```

Between 124.5 and 126.6 days into the year. Converting that into something we can understand (because I want to), there are $31 + 28 + 31 + 30 = 120$ days in January through April (in a non-leap year), so this says that the mean breakup date is between about May 4 and May 6.

The t-test is based on an assumption of data coming from a normal distribution. The histogram we made earlier looks pretty much normal, so there are no doubts about normality and thus no doubts about the validity of what we have done, on the evidence we have seen so far. (I have some doubts on different grounds, based on another of the plots we did earlier, which I'll explain later, but all I'm expecting you to do is to look at the histogram and say "Yep, that's normal enough". Bear in mind that the sample size is 87, which is large enough for the Central Limit Theorem to be pretty helpful, so that we don't need the data to be more than "approximately normal" for the sampling distribution of the sample mean to be very close to t with the right df.)

■

(c) An old-timer in Nenana strokes his grey beard and says "When I were young, I remember the tripod used to fall into the water around May 10". In a non-leap year, May 10 is Julian day 130. Test the null hypothesis that the mean `JulianDay` is 130, against the alternative that it is less. What do you conclude? What practical implication does that have (assuming that the old-timer has a good memory)?

Solution

The test is `t.test` again, but this time we have to specify a null mean and a direction of alternative:

```r
with(nenana, t.test(JulianDate, mu = 130, alternative = "less"))
```

```

    One Sample t-test

data:  JulianDate
t = -7.0063, df = 86, p-value = 2.575e-10
alternative hypothesis: true mean is less than 130
95 percent confidence interval:
     -Inf 126.6018
sample estimates:
mean of x
 125.5443
```

For a test, look first at the P-value, which is 0.0000000002575: that is to say, the P-value is very small, definitely smaller than 0.05 (or any other α you might have chosen). So we *reject* the null hypothesis, and conclude that the mean `JulianDate` is actually *less* than 130.

Now, this is the date on which the ice breaks up on average, and we have concluded that it is *earlier* than it used to be, since we are assuming the old-timer's memory is correct.

This is evidence in favour of global warming; a small piece of evidence, to be sure, but the ice is melting earlier than it used to all over the Arctic, so it's not just in Nenana that it is happening. You don't need to get to the "global warming" part, but I *do* want you to observe that the ice is breaking up earlier than it used to.

■

(d) Plot `JulianDate` against `Year` on a scatterplot. What recent trends, if any, do you see? Comment briefly. (You did this before, but I have some extra comments on the graph this time, so feel free to just read this part.)

Solution

I liked the `ggplot` with a smooth trend on it:

```r
ggplot(nenana, aes(x = Year, y = JulianDate)) + geom_point() + geom_smooth()
```

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

There was something obvious to see: after about 1960, there is a clear downward trend: the ice is breaking up earlier on average every year. Even though there is a lot of variability, the overall trend, viewed this way, is clear (and consistent with the test we did earlier). Note that the old-timer's value of 130 is the kind of `JulianDate` we would typically observe around 1920, which would make the old-timer over 90 years old.

All right, why did I say I had some doubts earlier? Well, because of this downward trend, the mean is not actually the same all the way through, so it doesn't make all that much sense to estimate it, which is what we were doing earlier by doing a confidence interval or a hypothesis test. What would actually make more sense is to estimate the mean `JulianDate` *for a particular year*. This could be done by a regression: predict `JulianDate` from `Year`, and then get a "confidence interval for the mean response" (as you would have seen in B27 or will see in C67). The trend isn't really linear, but is not that far off. I can modify the previous picture to give you an idea. Putting in `method="lm"` fits a line; as we see later, `lm` does regressions in R:

```
ggplot(nenana, aes(x = Year, y = JulianDate)) + geom_point() +
  geom_smooth(method = "lm")
```

```
`geom_smooth()` using formula = 'y ~ x'
```

Compare the confidence interval for the mean `JulianDate` in 1920: 126 to 131 (the shaded area on the graph), with 2000: 121 to 125. A change of about 5 days over 80 years. And with the recent trend that we saw above, it's probably changing faster than that now. Sobering indeed.

■

4.11 Diameters of trees

The Wade Tract in Thomas County, Georgia, is an old-growth forest of longleaf pine trees. It has survived in a relatively undisturbed state since before settlements of the area by Europeans. For each tree in the tract, researchers measured the diameter at breast height. This is a standard measure in forestry: it is defined as the diameter of the tree at 4.5 feet above the ground.[5] They are interested in the mean diameter at breast height of the trees in this tract. These values are in http://ritsokiguess.site/datafiles/treediameter.csv. The diameters are measured in centimetres. The easiest way to get the URL is to *right*-click on the blue text and select Copy URL. (If you copy and paste the actual text you might end up with extra spaces, especially if the printed URL goes over two lines.)

(a) Read in and display (some of) the data.

[5]The height of a typical human breast off the ground. Men have a breast too, you know.

Solution

The obvious way is this:

```
my_url <- "http://ritsokiguess.site/datafiles/treediameter.csv"
trees <- read_csv(my_url)
```

```
Rows: 40 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (1): diameter

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
trees
```

```
# A tibble: 40 x 1
   diameter
      <dbl>
 1     10.5
 2     13.3
 3     26
 4     18.3
 5     52.2
 6      9.2
 7     26.1
 8     17.6
 9     40.5
10     31.8
# i 30 more rows
```

Call the data frame what you like, though it is better to use a name that tells you what the dataframe contains (rather than something like `mydata`).

Extra 1: there is only one column, so you can pretend the columns are separated by anything at all. Thus you could use this:

```
my_url <- "http://ritsokiguess.site/datafiles/treediameter.csv"
trees <- read_delim(my_url, " ")
```

```
Rows: 40 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (1): diameter

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
trees
```

```
# A tibble: 40 x 1
   diameter
      <dbl>
 1     10.5
 2     13.3
 3     26
 4     18.3
 5     52.2
 6      9.2
 7     26.1
 8     17.6
 9     40.5
10     31.8
# i 30 more rows
```

or even this:

```r
my_url <- "http://ritsokiguess.site/datafiles/treediameter.csv"
trees <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  diameter = col_double()
)
```

```r
trees
```

```
# A tibble: 40 x 1
   diameter
      <dbl>
 1     10.5
 2     13.3
 3     26
 4     18.3
 5     52.2
 6      9.2
 7     26.1
 8     17.6
 9     40.5
10     31.8
# i 30 more rows
```

Extra 2: you might be wondering how they measure the diameter without doing something like drilling a hole through the tree. They don't actually measure the diameter at all. What they measure is the *circumference* of the tree, which is easy enough to do with a tape measure. Longleaf pines are usually near circular, so you get the diameter by taking the circumference and dividing by π. This City of Portland website shows you how it's done.

■

(b) Make a suitable plot of your dataframe.

Solution

One quantitative variable, so a histogram. Choose a sensible number of bins. There are 40 observations, so a number of bins up to about 10 is good. Sturges' rule says 6 since $2^6 = 64$:

```r
ggplot(trees, aes(x=diameter)) + geom_histogram(bins=6)
```

Extra 1: comments come later, but you might care to note (if only for yourself) that the distribution is a little skewed to the right, or, perhaps better, has *no* left tail at all. You might even observe that diameters cannot be less than 0 (they are measurements), and so you might expect a skew away from the limit.

After you've looked at the t procedures for these data, we'll get back to the shape.

Extra 2: later we look at a more precise tool for assessing normality, the normal quantile plot, which looks like this:

```
ggplot(trees, aes(sample=diameter)) + stat_qq() + stat_qq_line()
```

If the data come from a normal distribution, the points should follow the straight line, at least approximately. Here, most of the points do, except for the points on the left, which veer away upwards from the line: that is, the highest values, on the right, are about right for a normal distribution, but the lowest values, on the left, *don't go down low enough.*[6] Thus, the problem with normality is not the long tail on the right, but the short one on the left. It is hard to get this kind of insight from the histogram, but at the moment, it's the best we have.

The big problems, for things like t-tests that depend on means, is stuff like outliers, or long tails, with extreme values that might distort the mean. Having short tails, as the left tail here, will make the distribution look non-normal but won't cause any problems for the t-tests.

■

(c) Obtain a 95% confidence interval for the mean diameter.

Solution

This is `t.test`, but with `conf.level` to get the interval (and then you ignore the P-value):

```
with(trees, t.test(diameter))
```

```
One Sample t-test
```

[6]They cannot go down far enough, because they can't go below zero.

```
data:  diameter
t = 9.748, df = 39, p-value = 5.245e-12
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.6274 32.9526
sample estimates:
mean of x
    27.29
```

The mean diameter of a longleaf pine (like the ones in this tract) is between 21.6 and 33.0 centimetres.

If you prefer, do it this way:

```r
t.test(trees$diameter)
```

```
    One Sample t-test

data:  trees$diameter
t = 9.748, df = 39, p-value = 5.245e-12
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.6274 32.9526
sample estimates:
mean of x
    27.29
```

You need to *state the answer* and *round it off suitably.* The actual diameters in the data have one decimal place, so you can give the same accuracy for the CI, or *at most* two decimals (so 21.63 to 32.95 cm would also be OK).[7] Giving an answer with more decimals is something you cannot possibly justify. Worse even than giving too many decimals is not writing out the interval at all. *Never* make your reader find something in output. If they want it, tell them what it is.

Thus, here (if this were being graded), one mark for the output, one more for saying what the interval is, and the third if you give the interval with a sensible number of decimals.

■

[7] One more decimal place than the data is the maximum you give in a CI.

(d) Based on what you have seen so far, would you expect to reject a null hypothesis that the population mean diameter (of all longleaf pines like these) is 35 cm? Explain briefly. Then, carry out the test (against a two-sided alternative) and explain briefly whether you were right.

Solution

The logic is that "plausible" values for the population mean, ones you believe, are inside the interval, and implausible ones that you don't believe are outside. Remember that the interval is your best answer to "what is the population mean", and 35 is outside the interval so you don't think the population mean is 35, and thus you would reject it.

Are we right? Take out the `conf.level` and put in a `mu`:

```
with(trees, t.test(diameter, mu = 35))
```

```
    One Sample t-test

data:  diameter
t = -2.754, df = 39, p-value = 0.008895
alternative hypothesis: true mean is not equal to 35
95 percent confidence interval:
 21.6274 32.9526
sample estimates:
mean of x
    27.29
```

The P-value is less than our α of 0.05, so we would indeed reject a mean of 35 cm (in favour of the mean being different from 35).

■

(e) Would you expect 35 cm to be in a *99%* confidence interval for the mean diameter? Explain briefly, and then see if you were right.

Solution

The P-value is less than 0.01 (as well as being less than 0.05), so, in the same way that 35 was outside the 95% interval, it should be outside the 99% CI also. Maybe not by much, though, since the P-value is only just less than 0.01:

```
with(trees, t.test(diameter, conf.level = 0.99))
```

```
    One Sample t-test

data:  diameter
t = 9.748, df = 39, p-value = 5.245e-12
alternative hypothesis: true mean is not equal to 0
99 percent confidence interval:
 19.70909 34.87091
sample estimates:
mean of x
    27.29
```

Indeed so, outside, but only just.

■

4.12 One-sample cholesterol

The data set here contains cholesterol measurements for heart attack patients (at several different times) as well as for a group of control patients. We will focus on the control patients in this question.

(a) Read in and display (some of) the data.

Solution

This is (as you might guess) a `.csv`, so:

```r
my_url <- "http://ritsokiguess.site/datafiles/cholest.csv"
cholest <- read_csv(my_url)
```

```
Rows: 30 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (4): 2-Day, 4-Day, 14-Day, control

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
cholest
```

```
# A tibble: 30 x 4
   `2-Day` `4-Day` `14-Day` control
     <dbl>   <dbl>    <dbl>   <dbl>
 1     270     218      156     196
 2     236     234       NA     232
 3     210     214      242     200
 4     142     116       NA     242
 5     280     200       NA     206
 6     272     276      256     178
 7     160     146      142     184
 8     220     182      216     198
 9     226     238      248     160
10     242     288       NA     182
# i 20 more rows
```

Note for yourself that there are 30 observations (and some missing ones), and a column called `control` that is the one we'll be working with.

Extra: the 2-day, 4-day and 14-day columns need to be referred to with funny "backticks" around their names, because a column name cannot contain a `-` or start with a number. This is not a problem here, since we won't be using those columns, but if we wanted to, this would not work:

```
cholest %>% summarize(xbar = mean(2-Day))
```

```
Error in `summarize()`:
i In argument: `xbar = mean(2 - Day)`.
Caused by error in `h()`:
! error in evaluating the argument 'x' in selecting a method for function 'mean': object 'Day
```

because it is looking for a column called `Day`, which doesn't exist. The meaning of `2-Day` is "take the column called `Day` and subtract it from 2". To make this work, we have to supply the backticks ourselves:

```
cholest %>% summarize(xbar = mean(`2-Day`, na.rm = TRUE))
```

```
# A tibble: 1 x 1
   xbar
  <dbl>
1  254.
```

This column also has missing values (at the bottom), so here I've asked to remove the missing values[8] before working out the mean. Otherwise the mean is, unhelpfully, missing as well.

You might imagine that dealing with column names like this would get annoying. There is a package called `janitor` that has a function called `clean_names` to save you the trouble. Install it first, then load it:

```
library(janitor)
```

and then pipe your dataframe into `clean_names` and see what happens:

```
cholest %>% clean_names() -> cholest1
cholest1
```

```
# A tibble: 30 x 4
   x2_day x4_day x14_day control
    <dbl>  <dbl>   <dbl>   <dbl>
 1    270    218     156     196
 2    236    234      NA     232
 3    210    214     242     200
 4    142    116      NA     242
 5    280    200      NA     206
 6    272    276     256     178
 7    160    146     142     184
 8    220    182     216     198
 9    226    238     248     160
10    242    288      NA     182
# i 20 more rows
```

These are all legit column names; the `-` has been replaced by an underscore, and each of the first three column names has gained an `x` on the front so that it no longer starts with a number. This then works:

```
cholest1 %>% summarize(xbar = mean(x2_day, na.rm = TRUE))
```

```
# A tibble: 1 x 1
   xbar
  <dbl>
1  254.
```

[8]In R, missing values are labelled `NA`, and `rm` is Unix/C shorthand for *remove*.

■

(b) Make a suitable plot of the cholesterol levels of the control patients, and comment briefly on the shape of the distribution.

Solution

There is one quantitative variable, so a histogram, as ever:

```
ggplot(cholest, aes(x=control)) + geom_histogram(bins=6)
```

Pick a number of bins that shows the shape reasonably well. Too many or too few won't. (Sturges' rule says 6, since there are 30 observations and $2^5 = 32$.) Seven bins also works, but by the time you get to 8 bins or more, you are starting to lose a clear picture of the shape. Four bins is, likewise, about as low as you can go before getting too crude a picture.

Choosing one of these numbers of bins will make it clear that the distribution is somewhat skewed to the right.

■

(c) It is recommended that people in good health, such as the Control patients here, keep their cholesterol level below 200. Is there evidence that the mean cholesterol level of the population of people of which the Control patients are a sample is less than 200? Show that you understand the process, and state your conclusion in the context of the data.

Solution

The word "evidence" means to do a hypothesis test and get a P-value. Choose an α first, such as 0.05.

Testing a mean implies a one-sample t-test. We are trying to prove that the mean is less than 200, so that's our alternative: $H_a : \mu < 200$, and therefore the null is that the mean is equal to 200: $H_0 : \mu = 200$. (You might think it makes more logical sense to have $H_0 : \mu \geq 200$, which is also fine. As long as the null hypothesis has an equals in it in a logical place, you are good.)

```
with(cholest, t.test(control, mu=200, alternative = "less"))
```

```
    One Sample t-test

data:  control
t = -1.6866, df = 29, p-value = 0.05121
alternative hypothesis: true mean is less than 200
95 percent confidence interval:
     -Inf 200.0512
sample estimates:
mean of x
 193.1333
```

This is also good:

```
t.test(cholest$control, mu=200, alternative = "less")
```

```
    One Sample t-test

data:  cholest$control
t = -1.6866, df = 29, p-value = 0.05121
alternative hypothesis: true mean is less than 200
95 percent confidence interval:
     -Inf 200.0512
sample estimates:
mean of x
 193.1333
```

I like the first version better because a lot of what we do later involves giving a data frame, and then working with things in that data frame. This is more like that.

This test is *one*-sided because we are looking for evidence of *less*; if the mean is actually *more* than 200, we don't care about that. For a one-sided test, R requires you to say which side you are testing.

The P-value is not (quite) less than 0.05, so we cannot quite reject the null. Therefore, there is no evidence that the mean cholesterol level (of the people of which the control group are a sample) is less than 200. Or, this mean is not significantly less than 200. Or, we conclude that this mean is equal to 200. Or, we conclude that this mean could be 200. Any of those.

If you chose a different α, draw the right conclusion for the α you chose. For example, with $\alpha = 0.10$, we *do* have evidence that the mean is less than 200. Being consistent is more important than getting the same answer as me.

Writing out all the steps correctly shows that you understand the process. Anything less doesn't.

■

(d) What values could the population mean cholesterol level take? You might need to get some more output to determine this.

Solution

This is *not* quoting the sample mean, giving that as your answer, and then stopping. The sample mean should, we hope, be somewhere the population mean, but it is almost certainly not the same as the population mean, because there is variability due to random sampling. (This is perhaps the most important thing in all of Statistics: recognizing that variability exists and dealing with it.)

With that in mind, the question means to get a range of values that the population mean could be: that is to say, a confidence interval. The one that came out of the previous output is one-sided, to go with the one-sided test, but confidence intervals for us are two-sided, so we have to run the test again, but two-sided, to get it. To do that, take out the "alternative", thus (you can also take out the null mean, since a confidence interval has no null hypothesis):

```
with(cholest, t.test(control))
```

```
    One Sample t-test

data:  control
t = 47.436, df = 29, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
```

```
95 percent confidence interval:
 184.8064 201.4603
sample estimates:
mean of x
 193.1333
```

With 95% confidence, the population mean cholesterol level is between 184.8 and 201.5.

You need to state the interval, and you also need to round off the decimal places to something sensible. This is because in your statistical life, you are providing results to someone else *in a manner that they can read and understand.* They do not have time to go searching in some output, or to fish through some excessive number of decimal places. If that's what you give them, they will ask you to rewrite your report, wasting everybody's time when you could have done it right the first time.

How many decimal places is a good number? Look back at your data. In this case, the cholesterol values are whole numbers (zero decimal places). A confidence interval is talking about a mean. In this case, we have a sample size of 30, which is between 10 and 100, so we can justify one extra decimal place beyond the data, here one decimal altogether, or two *at the absolute outside.* (Two is more justifiable if the sample size is bigger than 100.) See, for example, this, in particular the piece at the bottom.

■

(e) Explain briefly why you would be reasonably happy to trust the t procedures in this question. (There are two points you need to make.)

Solution

The first thing is to look back at the graph you made earlier. This was skewed to the right ("moderately" or "somewhat" or however you described it). This would seem to say that the t procedures were not very trustworthy, since the population distribution doesn't look very normal in shape.

However, the second thing is to look at the sample size. We have the central limit theorem, which says (for us) that the larger the sample is, the less the normality matters, when it comes to estimating the mean. Here, the sample size is 30, which, for the central limit theorem, is large enough to overcome moderate non-normality in the data.

My take, which I was trying to guide you towards, is that our non-normality was not too bad, and so our sample size is large enough to trust the t procedures we used.

Extra 1: **There is nothing magical about a sample size of 30.** What matters is the tradeoff between sample size and the extent of the non-normality. If your data is less normal, you need a larger sample size to overcome it. Even a sample size of 500 might not be enough if your distribution is very skewed, or if you have extreme outliers.

The place $n = 30$ comes from is back from the days when we only ever used printed tables. In most textbooks, if you printed the t-table on one page in a decent-sized font, you'd get to about 29 df before running out of space. Then they would say "∞ df" and put the normal-distribution z numbers in. If the df you needed was bigger than what you had in the table, you used this last line: that is, you called the sample "large". Try it in your stats textbooks: I bet the df go up to 30, then you get a few more, then the z numbers.

Extra 2: By now you are probably thinking that this is very subjective, and so it is. What actually matters is the shape of the thing called the *sampling distribution of the sample mean*. That is to say, what kind of sample means you might get in repeated samples from your population. The problem is that you don't know what the population looks like.[9] But we can fake it up, in a couple of ways: we can play what-if and pretend we know what the population looks like (to get some understanding for "populations like that"), or we can use a technique called the "bootstrap" that will tell us what kind of sample means we might get from the population that *our* sample came from (this seems like magic and, indeed, is).

The moral of the story is that the central limit theorem is more powerful than you think.

To illustrate my first idea, let's pretend the population looks like this, with a flat top:

Only values between 0 and 1 are possible, and each of those is equally likely. Not very normal in shape. So let's take some random samples of size *three*, not in any sense a large sample, from this "uniform" population, and see what kind of sample means we get. This technique is called **simulation**: rather than working out the answer by math, we're letting the computer approximate the answer for us. Here's one simulated sample:

[9] If you did, all your problems would be over.

```r
u <- runif(3)
u
```

```
[1] 0.9475841 0.1245953 0.2277288
```

```r
mean(u)
```

```
[1] 0.4333027
```

and here's the same thing 1000 times, including a histogram of the sample means:

```r
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(runif(3))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 12)
```

This is our computer-generated assessment of what the sampling distribution of the sample mean looks like. Isn't this looking like a normal distribution?

Let's take a moment to realize what this is saying. If the population looks like the flat-topped uniform distribution, the central limit theorem kicks in for a sample of size *three*, and thus if your population looks like this, t procedures will be perfectly good for $n = 3$ or bigger, *even though the population isn't normal.*

Thus, when you're thinking about whether to use a t-test or something else (that we'll learn about later), the distribution shape matters, *but so does the sample size.*

I should say a little about my code. I'm not expecting you to figure out details now (we see the ideas properly in simulating power of tests), but in words, one line at a time:

- generate 1000 ("many") samples each of 3 observations from a uniform distribution
- for each sample, work out the mean of it
- turn those sample means into a data frame with a column called `value`
- make a histogram of those.

Now, the central limit theorem doesn't always work as nicely as this, but maybe a sample size of 30 is large enough to overcome the skewness that we had:

```
ggplot(cholest, aes(x=control)) + geom_histogram(bins=6)
```

That brings us to my second idea above.

The sample that we had is in some sense an "estimate of the population". To think about the sampling distribution of the sample mean, we need more estimates of the population. How

might we get those? The curious answer is to *sample from the sample*. This is the idea behind the *bootstrap*. (This is what Lecture 3c is about.) The name comes from the expression "pulling yourself up by your own bootstraps", meaning "to begin an enterprise or recover from a setback without any outside help" (from here), something that should be difficult or impossible. How is it possible to understand a sampling distribution with only one sample?

We have to be a bit careful. Taking a sample from the sample would give us the original sample back. So, instead, we sample *with replacement*, so that each bootstrap sample is different:

```
sort(cholest$control)
```

```
 [1] 160 162 164 166 170 176 178 178 182 182 182 182 182 184 186 188 196 198 198
[20] 198 200 200 204 206 212 218 230 232 238 242
```

```
sort(sample(cholest$control, replace=TRUE))
```

```
 [1] 164 166 166 166 166 176 178 178 182 182 182 182 182 188 198 198 198 200 200
[20] 200 200 204 206 206 218 218 230 232 232 242
```

A bootstrap sample contains repeats of the original data values, and misses some of the others. Here, the original data had values 160 and 162 that are missing in the bootstrap sample; the original data had one value 166, but the bootstrap sample has *four*! I sorted the data and the bootstrap sample to make this clearer; you will not need to sort. This is a perfectly good bootstrap sample:

```
sample(cholest$control, replace = TRUE)
```

```
 [1] 242 232 198 160 242 182 182 182 198 162 212 198 242 204 242 242 170 198 182
[20] 206 232 170 218 188 166 178 164 160 218 196
```

So now we know what to do: take lots of bootstrap samples, work out the mean of each, plot the means, and see how normal it looks. The only new idea here is the sampling with replacement:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(cholest$control, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 12)
```

That looks pretty normal, not obviously skewed, and so the t procedures we used will be reliable enough.

■

5 Two-sample inference

```
library(tidyverse)
```

5.1 Children and electronic devices

Do children (aged 8–17) spend more time on electronic devices now than they did 10 years ago? Samples of 15 children aged 8–17 were taken in each of two years, 1999 and 2009, and the children (with their parents' help) were asked to keep a diary of the number of hours they spent using electronic devices on a certain day. The data are in the file http://ritsokiguess.site/datafiles/pluggedin.txt.

(a) Read in the data and verify that you have 30 rows of data from two different years.

(b) Draw side-by-side boxplots of the number of hours for each year. `year` is a numeric variable that we want to treat as a factor, so we need to *make* it into a factor.

(c) Test whether the mean number of hours has *increased* since 1999. Which test did R do?

(d) Obtain a 99% confidence interval for the difference in means.

5.2 Parking close to the curb

In 2009, the Toronto Star commissioned a survey to address the issue of who is better at parking a car: men or women. The researchers recorded 93 drivers who were parallel-parking their car in downtown Toronto, and for each driver, recorded the distance between the car and the curb, in inches, when the driver was finished parking their car. The data are in an Excel spreadsheet, link. Click on the link. The data will probably download automatically. Check the folder on your computer where things get downloaded.[1] If the spreadsheet is just displayed and not downloaded, save it somewhere on your computer.

[1] Mine is rather prosaically called `Downloads`.

(a) There are two sheets in this spreadsheet workbook. They are of the same data in two different formats. Take a look at Sheet 1 and Sheet 2. Describe the format of the data in each case. Which will be the most suitable data layout, bearing in mind that one of the first things we do is to make side-by-side boxplots of parking distances for males and females? Explain briefly.

(b) Read your preferred sheet directly into R, *without* using a `.csv` file. (There is a clue in the lecture notes, in the section about reading in files.) If you get stuck, make a `.csv` file and read that in.

(c) Obtain side-by-side boxplots of parking distances for males and females. Does one gender seem to be better at parking than the other? Explain briefly.

(d) Explain briefly why this is two independent samples rather than matched pairs.

(e) Run a suitable t-test for comparing parking distances for males and females. What do you conclude, in the context of the data?

(f) Why might you have some doubts about the t-test that you just did? Explain briefly.

(g) The Toronto Star in its report said that females are more accurate at parking their cars. Why do you think they concluded that, and do you think they were right to do so? Explain briefly.

5.3 Bell peppers and too much water

A pathogen called *Phytophthora capsici* causes bell peppers to wilt and die. It is thought that too much water aids in the spread of the pathogen. Two fields are under study, labelled `a` and `b`. The first step in the research project is to compare the mean soil water content of the two fields. There is a suspicion that field `a` will have a higher water content than field `b`. The data are in the file link.

(a) Read the file in using `read_csv`, and list the resulting data frame.

(b) Make side-by-side boxplots of the water content values for the two fields. How do the fields seem to compare?

(c) Do a two-sample t-test to test whether there is evidence that the mean water content in field `a` is higher than that of field `b`. What do you conclude? Explain briefly. (You'll need to figure out a way of doing a one-sided test, or how to adapt the results from a two-sided test.)

(d) Is the result of your test consistent with the boxplot, or not? Explain briefly.

5.4 Exercise and anxiety and bullying mice

Does exercise help to reduce anxiety? To assess this, some researchers randomly assigned mice to either an enriched environment where there was an exercise wheel available, or a standard environment with no exercise options. After three weeks in the specified environment, for five minutes a day for two weeks, the mice were each exposed to a "mouse bully" — a mouse who was very strong, aggressive, and territorial. One measure of mouse anxiety is amount of time hiding in a dark compartment, with mice who are more anxious spending more time in darkness. The amount of time spent in darkness is recorded for each of the mice.

The data can be found at link.

(a) Read the data into R, and display your data frame. Count the number of mice in each group.

(b) Draw side-by-side boxplots of time spent in darkness for each group of mice.

(c) Do the boxplots support the hypothesis about exercise and anxiety? Explain briefly.

(d) Carry out a t-test for comparing the mean time spent in darkness for the mice in the two groups. Think carefully about the details of the t-test (and what you need evidence in favour of).

(e) What do you conclude, in terms of anxiety and exercise (at least for mice)? Explain briefly.

(f) Does anything in the previous parts suggest any problems with the analysis you just did? Explain briefly.

5.5 Diet and growth in boys

A dietician is studying the effect of different diets on children's growth. In part of the study, the dietician is investigating two religious sects, labelled `a` and `b` in our data set. Both sects are vegetarian; the difference between them is that people in Sect A only eat vegetables from below the ground, and Sect B only eats vegetables from above the ground. The height and weight of the boys[2] are measured at regular intervals. The data in link are the heights of the boys at age 12.

(a) Read in the data and find out how many observations you have and which variables.

(b) Obtain side-by-side boxplots of the heights for boys from each sect. Does it look as if the heights of the boys in each sect are different? Comment briefly.

[2]This was not sexism, but a recognition that boys and girls will be of different heights for reasons unrelated to diet. Doing it this way makes the analysis simpler.

(c) Looking at your boxplots, do you see any problems with doing a two-sample t-test? Explain briefly.

(d) Run a t-test to determine whether the mean heights differ significantly. What do you conclude? Explain briefly. (Run the t-test even if your previous work suggests that it is not the right thing to do.)

5.6 Handspans of males and females

Take your right hand, and stretch the fingers out as far as you can. The distance between the tip of your thumb and the tip of your little (pinky) finger is your handspan. The students in a Statistics class at Penn State measured their handspans and also whether they identified as male or female. The data are at http://ritsokiguess.site/datafiles/handspan.txt, with handspans measured in inches. Thinking of these as a random sample of all possible students, is it true that males have a larger mean handspan than females? This is what we will explore.

(a) Read in and display (some of) the data.

(b) Make a suitable graph of the two columns.

(c) Run a suitable two-sample t-test to address the question of interest. What do you conclude, in the context of the data?

(d) Obtain a 90% confidence interval for the difference in mean handspan between males and females. Do you need to run any more code? Explain briefly.

(e) Explain briefly why you might have some concerns about the validity of the t-tests you ran in this question. Or, if you don't have any concerns, explain briefly why that is.

5.7 The anchoring effect: Australia vs US

Two groups of students (in a class at a American university) were asked what they thought the population of Canada was. (The correct answer at the time was just over 30 million.) Some of the students, before having to answer this, were told that the population of the United States was about 270 million. The other students in the class were told that the population of Australia was about 18 million. The data are in http://ritsokiguess.site/datafiles/anchoring.csv. The first column contains the country whose population the student was told, and the second contains the student's guess at the population of Canada.

You might wonder how being told the population of an unrelated country would have any impact on a student's guess at the population of Canada. Psychology says it does: it's called the *anchoring effect*, and the idea is that the number mentioned first acts as an "anchor": a

person's guess will be closer to the anchor than it would have been otherwise. In this case, that would mean that the guesses for the students given the US as an anchor will be higher than for the students given Australia as an anchor. We are interested in seeing whether there is evidence for that here.

(a) Read in and display (some of) the data.

(b) Draw a suitable graph of these data.

(c) Explain briefly why a Welch t-test would be better than a pooled t-test in this case.

(d) Run a suitable Welch t-test and display the output.

(e) What do you conclude from your test, in the context of the data?

My solutions follow:

5.8 Children and electronic devices

Do children (aged 8–17) spend more time on electronic devices now than they did 10 years ago? Samples of 15 children aged 8–17 were taken in each of two years, 1999 and 2009, and the children (with their parents' help) were asked to keep a diary of the number of hours they spent using electronic devices on a certain day. The data are in the file http://ritsokiguess.site/datafiles/pluggedin.txt.

(a) Read in the data and verify that you have 30 rows of data from two different years.

Solution

I see this:

```
myurl="http://ritsokiguess.site/datafiles/pluggedin.txt"
plugged=read_delim(myurl," ")
```

```
Rows: 30 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): year, hours

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
plugged
```

```
# A tibble: 30 x 2
    year hours
   <dbl> <dbl>
 1  1999     4
 2  1999     5
 3  1999     7
 4  1999     7
 5  1999     5
 6  1999     7
 7  1999     5
 8  1999     6
 9  1999     5
10  1999     6
# i 20 more rows
```

I see only the first ten rows (with an indication that there are 20 more, so 30 altogether). In your notebook, it'll look a bit different: again, you'll see the first 10 rows, but you'll see exactly how many rows and columns there are, and there will be buttons "Next" and "Previous" to see earlier and later rows, and a little right-arrow to see more columns to the right (to which is added a little left-arrow if there are previous columns to scroll back to). If you want to check for yourself that there are 30 rows, you can click Next a couple of times to get down to row 30, and then see that the Next button cannot be clicked again, and therefore that 30 rows is how many there are.

Or, you can summarize the years by counting how many there are of each:

```
plugged %>% count(year)
```

```
# A tibble: 2 x 2
   year     n
  <dbl> <int>
1  1999    15
2  2009    15
```

or the more verbose form of the same thing:

```
plugged %>% group_by(year) %>% summarize(rows=n())
```

```
# A tibble: 2 x 2
   year  rows
  <dbl> <int>
```

```
1   1999     15
2   2009     15
```

Any of those says that it looks good. 30 rows, 1999 and 2009, 15 measurements for each.

■

(b) Draw side-by-side boxplots of the number of hours for each year. `year` is a numeric variable that we want to treat as a factor, so we need to *make* it into a factor.

Solution

```
ggplot(plugged,aes(x=factor(year),y=hours))+geom_boxplot()
```

The `fct_inorder` trick will also work, since the years are in the data in the order we want them to be displayed. (An option in case you have seen this.)

The median for 2009 is noticeably higher, and there is no skewness or outliers worth worrying about.

The measurements for the two years have a very similar spread, so there would be no problem running the pooled test here.

You might be bothered by the `factor(year)` on the x-axis. To get around that, you can define year-as-factor *first*, using `mutate`, then feed your new column into the boxplot. That goes like this. There is a wrinkle that I explain afterwards:

```
plugged %>% mutate(the_year=factor(year)) %>%
ggplot(aes(x=the_year, y=hours))+geom_boxplot()
```

You could even redefine `year` to be the factor version of itself (if you don't need the year-as-number anywhere else). The wrinkle I mentioned above is that in the `ggplot` you *do not* name the data frame first; the data frame used is the (nameless) data frame that came out of the previous step, not `plugged` but `plugged` with a new column `the_year`.

Note how the x-axis now has the name of the new variable.

If you forget to make `year` into a factor, this happens:

```
ggplot(plugged, aes(x = year, y = hours)) + geom_boxplot()
```

```
Warning: Continuous x aesthetic
i did you forget `aes(group = ...)`?
```

You get *one* boxplot, for all the hours, without distinguishing by year, and a warning message that tries (and fails) to read our mind: yes, we have a continuous, quantitative `x`, but `geom_boxplot` doesn't take a `group`. Or maybe it does. Try it and see:

```
ggplot(plugged,aes(x = year, y = hours, group = year)) + geom_boxplot()
```

The x-axis treats the year as a number, which looks a little odd, but adding a `group` correctly gets us two boxplots side by side, so this is also a good way to do it. So maybe the error message *did* read our mind after all.

■

(c) Test whether the mean number of hours has *increased* since 1999. Which test did R do?

Solution

The hard part to remember is how you specify a one-sided test in R; it's `alternative="less"` (rather than "greater") because 1999 is "before" 2009:

```
t.test(hours~year,data=plugged,alternative="less")
```

```
    Welch Two Sample t-test

data:  hours by year
t = -3.3323, df = 24.861, p-value = 0.001348
alternative hypothesis: true difference in means between group 1999 and group 2009 is less th
95 percent confidence interval:
       -Inf -0.8121415
sample estimates:
```

```
mean in group 1999 mean in group 2009
          5.933333           7.600000
```

The P-value is 0.0013. R does the Welch-Satterthwaite test by default (the unequal-variances one). Since we didn't change that, that's what we got. (The pooled test is below.)

This is the cleanest way to do it, because this version of `t.test`, with a "model formula" (the thing with the squiggle) allows a `data=` to say which data frame to get things from. The other ways, using (for example) `with`, also work:

```
with(plugged,t.test(hours~year,alternative="less"))
```

```
    Welch Two Sample t-test

data:  hours by year
t = -3.3323, df = 24.861, p-value = 0.001348
alternative hypothesis: true difference in means between group 1999 and group 2009 is less th
95 percent confidence interval:
       -Inf -0.8121415
sample estimates:
mean in group 1999 mean in group 2009
          5.933333           7.600000
```

This also works, but is *ugly*:

```
t.test(plugged$hours~plugged$year,alternative="less")
```

```
    Welch Two Sample t-test

data:  plugged$hours by plugged$year
t = -3.3323, df = 24.861, p-value = 0.001348
alternative hypothesis: true difference in means between group 1999 and group 2009 is less th
95 percent confidence interval:
       -Inf -0.8121415
sample estimates:
mean in group 1999 mean in group 2009
          5.933333           7.600000
```

Ugly because you've just typed the name of the data frame and the dollar sign *twice* for no reason. As a general principle, if you as a programmer are repeating yourself, you should stop and ask yourself how you can avoid the repeat.

If you want the pooled test in R, you have to ask for it:

```
t.test(hours~year, alternative = "less", data = plugged, var.equal = TRUE)
```

```
    Two Sample t-test

data:  hours by year
t = -3.3323, df = 28, p-value = 0.001216
alternative hypothesis: true difference in means between group 1999 and group 2009 is less th
95 percent confidence interval:
       -Inf -0.8158312
sample estimates:
mean in group 1999 mean in group 2009
          5.933333           7.600000
```

As is often the case, the P-values for the pooled and Welch-Satterthwaite tests are very similar, so from that point of view it doesn't matter much which one you use. If you remember back to the boxplots, the number of hours had about the same spread for the two years, so if you used the pooled test instead of the Welch-Satterthwaite test, that would have been just fine.

There is a school of thought that says we should learn the Welch-Satterthwaite test and use that always. This is because W-S (i) works when the populations from which the groups are sampled have different SDs and (ii) is pretty good even when those SDs are the same.

The pooled test can go badly wrong if the groups have very different SDs. The story is this: if the larger sample is from the population with the larger SD, the probability of a type I error will be smaller than α, and if the larger sample is from the population with the *smaller* SD, the probability of a type I error will be larger than α. This is why you see S-W in STAB22. You see the pooled test in STAB57 because the logic of its derivation is so much clearer, not because it's really the better test in practice. The theory says that if your data are normal in both groups with the same variance, then the pooled test is best, but it says *nothing* about the quality of the pooled test if any of that goes wrong. The usual approach to assessing things like this is via simulation, as we do for estimating power (later): generate some random data eg. from normal distributions with the same means, SDs 10 and 20 and sample sizes 15 and 30, run the pooled t-test, see if you reject, then repeat lots of times and see whether you reject about 5% of the time. Then do the same thing again with the sample sizes switched around. Or, do the same thing with Welch-Satterthwaite.

■

(d) Obtain a 99% confidence interval for the difference in means.

Solution

Take off the thing that made it one-sided, and put in a thing that gets the right CI:

```
t.test(hours~year,data=plugged,conf.level=0.99)
```

```
    Welch Two Sample t-test

data:  hours by year
t = -3.3323, df = 24.861, p-value = 0.002696
alternative hypothesis: true difference in means between group 1999 and group 2009 is not equ
99 percent confidence interval:
 -3.0614628 -0.2718705
sample estimates:
mean in group 1999 mean in group 2009
          5.933333           7.600000
```

-3.06 to -0.27. The interval contains only negative values, which is consistent with our having rejected a null hypothesis of no difference in means.

■

5.9 Parking close to the curb

In 2009, the Toronto Star commissioned a survey to address the issue of who is better at parking a car: men or women. The researchers recorded 93 drivers who were parallel-parking their car in downtown Toronto, and for each driver, recorded the distance between the car and the curb, in inches, when the driver was finished parking their car. The data are in an Excel spreadsheet, link. Click on the link. The data will probably download automatically. Check the folder on your computer where things get downloaded.[3] If the spreadsheet is just displayed and not downloaded, save it somewhere on your computer.

(a) There are two sheets in this spreadsheet workbook. They are of the same data in two different formats. Take a look at Sheet 1 and Sheet 2. Describe the format of the data in each case. Which will be the most suitable data layout, bearing in mind that one of the first things we do is to make side-by-side boxplots of parking distances for males and females? Explain briefly.

[3] Mine is rather prosaically called `Downloads`.

Solution

The data in Sheet 1 has one column of parking distances for males, and another for females. This is often how you see data of this sort laid out. Sheet 2 has *one* column of parking distances, all combined together, and a second column indicating the gender of the driver whose distance is in the first column. If you look back at the kind of data we've used to make side-by-side boxplots, it's always been in the format of Sheet 2: one column containing all the values of the variable we're interested in, with a second column indicating which group each observation belongs to ("group" here being "gender of driver"). So we need to use the data in Sheet 2, because the data in Sheet 1 are not easy to handle with R. The layout of Sheet 2 is the way R likes to do most things: so-called "long format" with a lot of rows and not many columns. This is true for descriptive stuff: side-by-side boxplots or histograms or means by group, as well as modelling such as (here) a two-sample t-test, or (in other circumstances, as with several groups) a one-way analysis of variance. Hadley Wickham, the guy behind the `tidyverse`, likes to talk about "tidy data" (like Sheet 2), with each column containing a variable, and "untidy data" (like Sheet 1), where the two columns are the same thing (distances), but under different circumstances (genders). As we'll see later, it is possible to convert from one format to the other. Usually you want to make untidy data tidy (the function for this is called `pivot_longer`).

■

(b) Read your preferred sheet directly into R, *without* using a `.csv` file. (There is a clue in the lecture notes, in the section about reading in files.) If you get stuck, make a `.csv` file and read that in.

Solution

The direct way is to use the package `readxl`. This has a `read_excel` that works the same way as any of the other `read_` functions. You'll have to make sure that you read in sheet 2, since that's the one you want. There is some setup first. There are a couple of ways you can do that:

- Download the spreadsheet to your computer, and upload it to your project on R Studio Cloud (or, if you are running R Studio on your computer, use something like `file.choose` to get the file from wherever it got downloaded to).
- Use the function `download.file` to get the file from the URL and store it in your project folder directly. This also works in R Studio Cloud, and completely by-passes the download-upload steps that you would have to do otherwise. (I am grateful to Rose Gao for this idea.) Here is how you can use `download.file` here:

```
my_url <- "http://ritsokiguess.site/datafiles/parking.xlsx"
local <- "parking.xlsx"
download.file(my_url, local, mode = "wb")
```

When you've gotten the spreadsheet into your project folder via one of those two ways, you go ahead and do this:

```
library(readxl)
parking <- read_excel("parking.xlsx", sheet = 2)
parking
```

```
# A tibble: 93 x 2
   distance gender
      <dbl> <chr>
 1      0.5 male
 2      1   male
 3      1.5 male
 4      1.5 male
 5      1.5 male
 6      3   male
 7      3.5 male
 8      5   male
 9      6   male
10      6   male
# i 83 more rows
```

You have to do it this way, using the version of the spreadsheet on your computer, since `read_excel` won't take a URL, or if it does, I can't make it work.[4] I put the spreadsheet in R Studio's current folder, so I could read it in by name, or you can do the `f <- file.choose()` thing, find it, then read it in. The `sheet=` thing can take either a number (as here: the second sheet in the workbook), or a name (whatever name the sheet has on its tab in the workbook).

Extra: Rose actually came up with a better idea, which I will show you and explain:

```
tf <- tempfile()
download.file(my_url, tf, mode = "wb")
p <- read_excel(tf, sheet = 2)
```

What `tempfile()` does is to create a temporary file to hold the spreadsheet that you are about to download. After downloading the spreadsheet to the temporary file, you then use `read_excel` to read *from the temporary file* into the data frame.

The advantage of this approach is that the temporary file disappears as soon as you close R, and so you don't have a copy of the spreadsheet lying around that you don't need (once you have created the dataframe that I called `parking`, anyway).

[4] Let me know if you have more success than I did.

If you are wondering about that `mode` thing on `download.file`: files are of two different types, "text" (like the text of an email, that you can open and look at in something like Notepad), and "binary" that you can't look at directly, but for which you need special software like Word or Excel to decode it for you.[5]
The first character in `mode` is either `w` for "write a new file", which is what we want here, or `a` for "append", which would mean adding to the end of a file that already exists. Thus `mode="wb"` means "create a new binary file". End of Extra.

If you can't make any of this work, then do it in two steps: save the appropriate sheet as a `.csv` file, and then read the `.csv` file using `read_csv`. If you experiment, you'll find that saving a spreadsheet workbook as `.csv` only saves the sheet you're looking at, so make sure you are looking at sheet 2 before you Save As `.csv`. I did that, and called my saved `.csv` `parking2.csv` (because it was from sheet 2, but you can use any name you like). Then I read this into R thus:

```
parking2 <- read_csv("parking2.csv")
```

```
Rows: 93 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): gender
dbl (1): distance

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
parking2
```

```
# A tibble: 93 x 2
   distance gender
      <dbl> <chr>
 1      0.5 male
 2      1   male
 3      1.5 male
 4      1.5 male
 5      1.5 male
 6      3   male
 7      3.5 male
 8      5   male
```

[5]A Word or Excel document has all kinds of formatting information hidden in the file as well as the text that you see on the screen.

```
 9      6   male
10      6   male
# i 83 more rows
```

The read-in data frame `parking` has 93 rows ($47 + 46 = 93$ drivers) and two columns: the distance from the curb that the driver ended up at, and the gender of the driver. This is as the spreadsheet Sheet 2 was, and the first few distances match the ones in the spreadsheet.

If I were grading this, you'd get some credit for the `.csv` route, but I really wanted you to figure out how to read the Excel spreadsheet directly, so that's what would be worth full marks.

You might want to check that you have some males and some females, and how many of each, which you could do this way:

```
parking %>% count(gender)
```

```
# A tibble: 2 x 2
  gender     n
  <chr>  <int>
1 female    47
2 male      46
```

■

(c) Obtain side-by-side boxplots of parking distances for males and females. Does one gender seem to be better at parking than the other? Explain briefly.

Solution

With the right data set, this is a piece of cake:

```
ggplot(parking, aes(x = gender, y = distance)) + geom_boxplot()
```

The outcome variable is distance from the curb, so smaller should be better (more accurate parking). With that in mind, the median for females is a little smaller than for males (about 8.5 vs. about 10), so it seems that on average females are more accurate parkers than males are. The difference is small, however (and so you might be wondering at this point whether it's a statistically significant difference — don't worry, that's coming up).

Before I leave this one, I want to show you something else: above-and-below histograms, as another way of comparing males and females (two or more groups, in general). First, we make a histogram of all the distances, without distinguishing by gender:

```
ggplot(parking, aes(x = distance)) + geom_histogram(bins = 8)
```

That big outlier is the very inaccurate male driver.

Now, how do we get a *separate* histogram for each `gender`? In `ggplot`, separate plots for each "something" are called **facets**, and the way to get facets arranged as you want them is called `facet_grid`.[6] Let me show you the code first, and then explain how it works:

```
ggplot(parking, aes(x = distance)) +
  geom_histogram(bins = 7) +
  facet_grid(gender ~ .)
```

[6] I wrote this question a long time ago, back when I thought that `facet_grid` was the only way to do facets. Now, I would use `facet_wrap`. See the discussion about `facet_wrap` near the bottom.

`facet_grid` takes a "model formula" with a squiggle, with y on the left and x on the right. We want to compare our two histograms, one for males and one for females, and I think the best way to compare histograms is to have one on top of the other. Note that the same `distance` scale is used for both histograms, so that it is a fair comparison. The above-and-below is accomplished by having `gender` as the y in the arrangement of the facets, so it goes before the squiggle. We don't have any x in the arrangement of the facets, and we tell `ggplot` this by putting a dot where the x would be.[7]

You can also use `facet_wrap` for this, but you have to be more careful since you don't have any control over how the histograms come out (you probably get them side by side, which is not so helpful for comparing distributions). You can make it work by using `ncol=1` to arrange *all* the histograms in one column:

```
ggplot(parking, aes(x = distance)) +
  geom_histogram(bins = 7) +
  facet_wrap(~gender, ncol = 1)
```

[7]You might have a second categorical variable by which you want to arrange the facets left and right, and that would go where the dot is.

The centres of both histograms are somewhere around 10, so it's hard to see any real difference between males and females here. Maybe this is further evidence that the small difference we saw between the boxplots is really not worth getting excited about.

You might be concerned about how you know what to put with the squiggle-thing in `facet_grid` and `facet_wrap`. The answer is that `facet_wrap` *only* has something to the right of the squiggle (which `ggplot` then decides how to arrange), but `facet_grid` *must* have something on *both* sides of the squiggle (how to arrange in the y direction on the left, how to arrange in the x direction on the right), and if you don't have anything else to put there, you put a dot. Here's my `facet_grid` code from above, again:

```
ggplot(parking, aes(x = distance)) +
  geom_histogram(bins = 7) +
  facet_grid(gender ~ .)
```

We wanted gender to go up and down, and we had nothing to go left and right, hence the dot. Contrast that with my `facet_wrap` code:[8]

```
ggplot(parking, aes(x = distance)) +
  geom_histogram(bins = 7) +
  facet_wrap(~gender)
```

[8] I took out the `ncol` since that confuses the explanation here.

This says "make a separate facet for each gender", but it doesn't say anything about how to arrange them. The choice of bins for my histogram(s) came from Sturges' rule: with n being the number of observations, you use k bins where $k = \log_2(n) + 1$, rounded up. If we were to make a histogram of all the parking distances combined together, we would have $n = 47 + 48 = 95$ observations, so we should use this many bins:

```
sturges <- log(95, 2) + 1
sturges
```

```
[1] 7.569856
```

Round this up to 8. (The second thing in `log` is the base of the logs, if you specify it, otherwise it defaults to e and gives you "natural" logs.) I seem to have the powers of 2 in my head, so I can do it mentally by saying "the next power of 2 is 128, which is 2^7, so I need $7 + 1 = 8$ bins."

Or:

```
with(parking, nclass.Sturges(distance))
```

```
[1] 8
```

Sturges' rule tends to produce not enough bins if n is small, so be prepared to increase it a bit if you don't have much data. I think that gives a fairly bare-bones picture of the shape: skewed to the right with outlier.

The other rule we saw was Freedman-Diaconis:

```
with(parking, nclass.FD(distance))
```

```
[1] 14
```

and that leads to this histogram:

```
ggplot(parking, aes(x = distance)) + geom_histogram(bins = 14)
```

That gives rather more detail (a lot more bars: the binwidth in the Sturges-rule histogram is about 7, or twice what you see here), but in this case the overall story is about the same.

In the case of faceted histograms, you would want to apply a rule that uses the number of observations in *each* histogram. The facets might have quite different numbers of observations, but you can only use one `binwidth` (or `bins`), so you may have to compromise. For example, using Sturges' rule based on 47 observations (the number of males; the number of females is one more):

```
log(47, 2) + 1
```

```
[1] 6.554589
```

and so each facet should have that many bins, rounded up. That's where I got my 7 for the facetted histogram from. This one doesn't work immediately with `nclass.Sturges`, because we do not have *one* column whose length is the number of observations we want: we have one column of distances that are males and females mixed up. To do *that*, `filter` one of the genders first:

```
parking %>%
  filter(gender == "female") %>%
  with(., nclass.Sturges(distance))
```

```
[1] 7
```

I used the "dot" trick again, which you can read as "it": "from `parking`, take only the rows for the females, and with it, give me the number of bins for a histogram by Sturges' rule."

■

(d) Explain briefly why this is two independent samples rather than matched pairs.

Solution

There is no way to pair any male with a corresponding female, because they are unrelated people. You might also notice that there are not even the *same number* of males and females, so there can be no way of pairing them up without leaving one over. (In general, if the two samples are paired, there must be the same number of observations in each; if there are different numbers in each, as here, they cannot be paired.) If you want that more mathematically, let n_1 and n_2 be the two sample sizes; then:

$$\text{Paired} \Longrightarrow n_1 = n_2$$

from which it follows logically (the "contrapositive") that

$$n_1 \neq n_2 \Longrightarrow \text{not paired}$$

You'll note from the logic that if the two sample sizes are the same, that tells you *nothing* about whether it's paired or independent samples: it could be either, and in that case you have to look at the description of the data to decide between them.

Here, anything that gets at why the males and females cannot be paired up is good.

■

(e) Run a suitable t-test for comparing parking distances for males and females. What do you conclude, in the context of the data?

Solution

A two-sample t-test. I think either the Welch or the pooled one can be justified (and I would expect them to give similar answers). You can do the Welch one either without comment or by asserting that the boxplots show different spreads; if you are going to do the pooled one, you need to say that the spreads are "about equal", by comparing the heights of the boxes on the boxplots:

```r
t.test(distance ~ gender, data = parking)
```

```
    Welch Two Sample t-test

data:  distance by gender
t = -1.3238, df = 79.446, p-value = 0.1894
alternative hypothesis: true difference in means between group female and group male is not e
95 percent confidence interval:
 -4.5884103  0.9228228
sample estimates:
mean in group female   mean in group male
            9.308511            11.141304
```

This is the Welch-Satterthwaite version of the test, the one that does not assume equal SDs in the two groups. The P-value of 0.1894 is not small, so there is no evidence of any difference in parking accuracy between males and females.

Or, this being the pooled one:

```
t.test(distance ~ gender, data = parking, var.equal = TRUE)
```

```
    Two Sample t-test

data:  distance by gender
t = -1.329, df = 91, p-value = 0.1872
alternative hypothesis: true difference in means between group female and group male is not e
95 percent confidence interval:
 -4.5722381  0.9066506
sample estimates:
mean in group female   mean in group male
            9.308511            11.141304
```

You might have thought, looking at the boxplots, that the groups had about the same SD (based, for example, on noting that the two boxes were about the same height, so the IQRs were about the same). In that case, you might run a pooled t-test, which here gives an almost identical P-value of 0.1872, and the exact same conclusion.

■

(f) Why might you have some doubts about the t-test that you just did? Explain briefly.

Solution

The two-sample t-test is based on an assumption of normally-distributed data within each group. If you go back and look at the boxplots, you'll see either (depending on your point of view) that both groups are right-skewed, or that both groups have outliers, neither of which fits a normal distribution. The outlier in the male group is particularly egregious.[9] So I think we are entitled to question whether a two-sample t-test is the right thing to do. Having said that, we should go back and remember that the t-tests are "robust to departures from normality" (since we are working with the Central Limit Theorem here), and therefore that this test might be quite good even though the data are not normal, because the sample sizes of 40-plus are large (by the standards of what typically makes the Central Limit Theorem work for us). So it may not be as bad as it seems. A common competitor for the two-sample t-test is the Mann-Whitney test. This doesn't assume normality, but it *does* assume symmetric distributions, which it's not clear that we have here. I like a test called Mood's Median Test, which is kind of the two-sample equivalent of the sign test (which we will also see later). It goes like this: Work out the overall median of all the distances, regardless of gender:

```
parking %>% summarize(med = median(distance))
```

```
# A tibble: 1 x 1
    med
  <dbl>
1     9
```

The overall median is 9.

Count up how many distances of each gender were above or below the overall median. (Strictly, I'm supposed to throw away any values that are exactly equal to the overall median, but I won't here for clarity of exposition.)

```
tab <- with(parking, table(gender, distance < 9))
tab
```

```
gender   FALSE TRUE
  female    23   24
  male      27   19
```

For example, 19 of the male drivers had a distance (strictly) less than 9. Both genders are pretty close to 50–50 above and below the overall median, which suggests that the males

[9] Google defines this as meaning "outstandingly bad, shocking".

and females have about the same median. This can be tested (it's a chi-squared test for independence, if you know that):

```
chisq.test(tab, correct = F)
```

```
    Pearson's Chi-squared test

data:  tab
X-squared = 0.89075, df = 1, p-value = 0.3453
```

This is even less significant (P-value 0.3453) than the two-sample t-test, and so is consistent with our conclusion from before that there is actually no difference between males and females in terms of average parking distance. The Mood's median test is believable because it is not affected by outliers or distribution shape.

■

(g) The Toronto Star in its report said that females are more accurate at parking their cars. Why do you think they concluded that, and do you think they were right to do so? Explain briefly.

Solution

The conclusion from the boxplots was that the female median distance was less than the males, slightly, in this sample. That is probably what the Star seized on. Were they right? Well, that was why we did the test of significance. We were trying to see whether this observed difference between males and females was "real" (would hold up if you looked at "all" male and female drivers) or "reproducible" (you would expect to see it again if you did another study like this one). The large, non-significant P-values in all our tests tell us that the difference observed here was nothing more than chance. So it was not reasonable to conclude that females generally are more accurate at parallel-parking than males are.

■

5.10 Bell peppers and too much water

A pathogen called *Phytophthora capsici* causes bell peppers to wilt and die. It is thought that too much water aids in the spread of the pathogen. Two fields are under study, labelled `a` and `b`. The first step in the research project is to compare the mean soil water content of the two fields. There is a suspicion that field `a` will have a higher water content than field `b`. The data are in the file link.

(a) Read the file in using `read_csv`, and list the resulting data frame.

Solution

Reading directly from the URL is easiest:

```
my_url <- "http://ritsokiguess.site/datafiles/bellpepper.csv"
pepper <- read_csv(my_url)
```

```
Rows: 30 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): field
dbl (1): water

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pepper
```

```
# A tibble: 30 x 2
   field water
   <chr> <dbl>
 1 a      10.2
 2 a      10.7
 3 a      15.5
 4 a      10.4
 5 a       9.9
 6 a      10
 7 a      16.6
 8 a      15.1
 9 a      15.2
10 a      13.8
# i 20 more rows
```

If you like, find out how many observations you have from each field, thus:

```
pepper %>% count(field)
```

```
# A tibble: 2 x 2
  field     n
  <chr> <int>
1 a        14
2 b        16
```

Fourteen and sixteen.

■

(b) Make side-by-side boxplots of the water content values for the two fields. How do the fields seem to compare?

Solution

This kind of thing:

```
ggplot(pepper, aes(x = field, y = water)) + geom_boxplot()
```

This one is rather interesting: the distribution of water contents for field `a` is generally higher than that for field `b`, but the median for `a` is actually *lower*.

The other reasonable plot is a facetted histogram, something like this:

```r
ggplot(pepper, aes(x = water)) + geom_histogram(bins = 6) +
  facet_grid(field ~ .)
```

The distribution of water content in field `b` is actually bimodal, which is probably the explanation of the funny thing with the median. What actually seems to be happening (at least for these data) is that the water content in field B is either about the same as field A, or a lot less (nothing in between). I can borrow an idea from earlier to find the five-number summaries for each field:

```r
pepper %>%
  nest(-field) %>%
  rowwise() %>%
  mutate(qq = list(enframe(quantile(data$water))))%>%
  unnest(qq) %>%
  select(-data) %>%
  pivot_wider(names_from=name, values_from=value)
```

```
Warning: Supplying `...` without names was deprecated in tidyr 1.0.0.
i Please specify a name for each selection.
i Did you want `data = -field`?

# A tibble: 2 x 6
```

```
  field  `0%` `25%` `50%` `75%` `100%`
  <chr> <dbl> <dbl> <dbl> <dbl>  <dbl>
1 a       9.9 10.5   11.4  14.8   16.6
2 b       7.1  8.33  11.8  12.6   13.9
```

This is a weird one: all the quantiles are greater for field A *except* for the median.

■

(c) Do a two-sample t-test to test whether there is evidence that the mean water content in field `a` is higher than that of field `b`. What do you conclude? Explain briefly. (You'll need to figure out a way of doing a one-sided test, or how to adapt the results from a two-sided test.)

Solution

```r
t.test(water ~ field, alternative = "greater", data = pepper)
```

```
    Welch Two Sample t-test

data:  water by field
t = 2.0059, df = 27.495, p-value = 0.0274
alternative hypothesis: true difference in means between group a and group b is greater than
95 percent confidence interval:
 0.2664399       Inf
sample estimates:
mean in group a mean in group b
       12.52857        10.76875
```

Note the use of `alternative` to specify that the first group mean (that of field `a`) is bigger than the second, field `b`, under the alternative hypothesis.

The P-value, 0.0274, is less than 0.05, so we reject the null (equal means) in favour of the `a` mean being bigger than the `b` mean: field `a` really does have a higher mean water content.

Another way to tackle this is to do a two-sided test and adapt the P-value:

```r
t.test(water ~ field, data = pepper)
```

```
    Welch Two Sample t-test

data:  water by field
t = 2.0059, df = 27.495, p-value = 0.0548
alternative hypothesis: true difference in means between group a and group b is not equal to
95 percent confidence interval:
 -0.03878411  3.55842696
sample estimates:
mean in group a mean in group b
       12.52857        10.76875
```

This time we do *not* go straight to the P-value. First we check that we are on the correct side, which we are since the sample mean for field `a` *is* bigger than for field `b`. Then we are entitled to take the two-sided P-value 0.0548 and *halve* it to get the same 0.0274 that we did before.

■

(d) Is the result of your test consistent with the boxplot, or not? Explain briefly.

Solution

The test said that field `a` had a greater mean water content. Looking at the boxplot, this is consistent with where the boxes sit (`a`'s box is higher up than `b`'s). However, it is not consistent with the medians, where `b`'s median is actually *bigger*. You have two possible right answers here: comparing the boxes with the test result (they agree) or comparing the medians with the test result (they disagree). Either is good. If you like, you could also take the angle that the two boxes overlap a fair bit, so it is surprising that the test came out significant. (The resolution of this one is that we have 30 measurements altogether, 14 and 16 in the two groups, so the sample size is not tiny. With smaller samples, having overlapping boxes would probably lead to a non-significant difference.)

■

5.11 Exercise and anxiety and bullying mice

Does exercise help to reduce anxiety? To assess this, some researchers randomly assigned mice to either an enriched environment where there was an exercise wheel available, or a standard environment with no exercise options. After three weeks in the specified environment, for five minutes a day for two weeks, the mice were each exposed to a "mouse bully" — a mouse who was very strong, aggressive, and territorial. One measure of mouse anxiety is amount of time hiding in a dark compartment, with mice who are more anxious spending more time in darkness. The amount of time spent in darkness is recorded for each of the mice.

The data can be found at link.

(a) Read the data into R, and display your data frame. Count the number of mice in each group.

Solution

These are aligned columns with spaces in between, so we need `read_table`:

```
my_url <- "http://ritsokiguess.site/datafiles/stressedmice.txt"
mice <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  Time = col_double(),
  Environment = col_character()
)
```

```
mice
```

```
# A tibble: 14 x 2
    Time Environment
   <dbl> <chr>
 1   359 Enriched
 2   280 Enriched
 3   138 Enriched
 4   227 Enriched
 5   203 Enriched
 6   184 Enriched
 7   231 Enriched
 8   394 Standard
 9   477 Standard
10   439 Standard
11   428 Standard
12   391 Standard
13   488 Standard
14   454 Standard
```

You can call the data frame whatever you like.

If you must, you can physically count the number of mice in each group, but you ought to get in the habit of coding this kind of thing:

```r
mice %>% count(Environment)
```

```
# A tibble: 2 x 2
  Environment     n
  <chr>       <int>
1 Enriched        7
2 Standard        7
```

Seven in each.

■

(b) Draw side-by-side boxplots of time spent in darkness for each group of mice.

Solution

This:

```r
ggplot(mice, aes(x = Environment, y = Time)) + geom_boxplot()
```

You did remember to put capital letters on the variable names, didn't you?

■

(c) Do the boxplots support the hypothesis about exercise and anxiety? Explain briefly.

Solution

The hypothesis about exercise and anxiety is that mice who exercise more should be less anxious. How does that play out in this study? Well, mice in the enriched environment at least have the opportunity to exercise, which the mice in the standard environment do not, and anxiety is measured by the amount of time spent in darkness (more equals more anxious). So we'd expect the mice in the standard environment to spend more time in darkness, if that hypothesis is correct. That's exactly what the boxplots show, with very little doubt.[10] Your answer needs to make two points: (i) what you would expect to see, if the hypothesis about anxiety and exercise is true, and (ii) whether you actually did see it. You can do this either way around: for example, you can say what you see in the boxplot, and then make the case that this *does* support the idea of more exercise corresponding with less anxiety.

■

(d) Carry out a t-test for comparing the mean time spent in darkness for the mice in the two groups. Think carefully about the details of the t-test (and what you need evidence in favour of).

Solution

We are trying to prove that exercise goes with *less* anxiety, so a one-sided test is called for. The other thing to think about is how R organizes the groups for `Environment`: in alphabetical order. Thus `Enriched` is first (like on the boxplot). We're trying to prove that the mean `Time` is *less* for `Enriched` than for `Standard`, so we need `alternative="less"`:

```
with(mice, t.test(Time ~ Environment, alternative = "less"))
```

```
    Welch Two Sample t-test

data:  Time by Environment
t = -6.7966, df = 9.1146, p-value = 3.734e-05
alternative hypothesis: true difference in means between group Enriched and group Standard is
95 percent confidence interval:
      -Inf -151.2498
sample estimates:
mean in group Enriched mean in group Standard
              231.7143               438.7143
```

[10]This means that I would expect to reject a null hypothesis of equal means, but I get ahead of myself.

A common clue that you have the wrong alternative hypothesis is a P-value coming out close to *1*, which is what you would have gotten from something like this:

```
with(mice, t.test(Time ~ Environment, alternative = "greater"))
```

```
    Welch Two Sample t-test

data:  Time by Environment
t = -6.7966, df = 9.1146, p-value = 1
alternative hypothesis: true difference in means between group Enriched and group Standard is
95 percent confidence interval:
 -262.7502       Inf
sample estimates:
mean in group Enriched mean in group Standard
              231.7143               438.7143
```

Here, we looked at the pictures and expected to find a difference, so we expected to find a P-value close to 0 rather than close to 1.

■

(e) What do you conclude, in terms of anxiety and exercise (at least for mice)? Explain briefly.

Solution

The P-value (from the previous part) is 0.000037, which is way less than 0.05 (or 0.01 or whatever α you chose). So the null hypothesis (equal means) is resoundingly rejected in favour of the one-sided alternative that the mean anxiety (as measured by time spent in darkness) is less for the mice who (can) exercise. You need to end up by doing a one-sided test. An alternative to what I did is to do a two-sided test in the previous part. Then you can fix it up by recognizing that the means are the right way around for the research hypothesis (the mean time in darkness is way less for `Enriched`), and then dividing the two-sided P-value by 2. But you need to do the "correct side" thing: just halving the two-sided P-value is not enough, because the sample mean for `Enriched` might have been *more* than for `Standard`.

■

(f) Does anything in the previous parts suggest any problems with the analysis you just did? Explain briefly.

Solution

Look at the side-by-side boxplots. The strict assumptions hiding behind the t-tests are that the data in each group come from normal distributions (equal standard deviations are not required). Are the data symmetric? Are there any outliers? Well, I see a high outlier in the `Enriched` group, so I have some doubts about the normality. On the other hand, I only have seven observations in each group, so there is no guarantee even if the populations from which they come are normal that the samples will be. So maybe things are not so bad. This is one of those situations where you make a case and defend it. I don't mind so much which case you make, as long as you can defend it. Thus, something like either of these two is good:

- I see an outlier in the `Enriched` group. The data within each group are supposed to be normally distributed, and the `Enriched` group is not. So I see a problem.
- I see an outlier in the `Enriched` group. But the sample sizes are small, and an apparent outlier could arise by chance. So I do not see a problem.

Extra: another way to think about this is normal quantile plots to assess normality within each group. This uses the facetting trick to get a separate normal quantile plot for each `Environment`:

```
ggplot(mice, aes(sample = Time)) + stat_qq() + stat_qq_line() +
  facet_wrap(~Environment, scales = "free")
```

For the `Enriched` group, the upper-end outlier shows up. In a way this plot is no more illuminating than the boxplot, because you still have to make a call about whether this is "too big". Bear in mind also that these facetted normal quantile plots, with two groups, come out tall and skinny, so vertical deviations from the line are exaggerated. On this plot, the lowest value also looks too low.

For the `Standard` group, there are no problems with normality at all.

What happens if we change the shape of the plots?

```
ggplot(mice, aes(sample = Time)) + stat_qq() + stat_qq_line() +
  facet_wrap(~Environment, scales = "free", ncol = 1)
```

This makes the plots come out in one column, that is, short and squat. I prefer these. I'd still call the highest value in `Enriched` an outlier, but the lowest value now looks pretty close to what you'd expect.

■

5.12 Diet and growth in boys

A dietician is studying the effect of different diets on children's growth. In part of the study, the dietician is investigating two religious sects, labelled `a` and `b` in our data set. Both sects

are vegetarian; the difference between them is that people in Sect A only eat vegetables from below the ground, and Sect B only eats vegetables from above the ground. The height and weight of the boys[11] are measured at regular intervals. The data in link are the heights of the boys at age 12.

(a) Read in the data and find out how many observations you have and which variables.

Solution

The data values are separated by one space, so:

```
my_url <- "http://ritsokiguess.site/datafiles/kids-diet.txt"
diet <- read_delim(my_url, " ")
```

```
Rows: 21 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): sect
dbl (1): height

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
diet
```

```
# A tibble: 21 x 2
   sect  height
   <chr>  <dbl>
 1 a        140
 2 a        140
 3 a        140
 4 a        143
 5 a        135
 6 a        144
 7 a        156
 8 a        149
 9 a        146
10 a        148
# i 11 more rows
```

[11]This was not sexism, but a recognition that boys and girls will be of different heights for reasons unrelated to diet. Doing it this way makes the analysis simpler.

21 observations on two variables, `sect` and `height`. (You should state this; it is not enough to make the reader figure it out for themselves.)

The heights are evidently in centimetres.

You can call the data frame whatever you like.

■

(b) Obtain side-by-side boxplots of the heights for boys from each sect. Does it look as if the heights of the boys in each sect are different? Comment briefly.

Solution

The boxplot is the kind of thing we've seen before:

```
ggplot(diet, aes(x = sect, y = height)) + geom_boxplot()
```

It looks to me as if the boys in Sect B are taller on average.

■

(c) Looking at your boxplots, do you see any problems with doing a two-sample t-test? Explain briefly.

Solution

The assumption is that the data in each group are "approximately normal". Boxplots don't tell you about normality specifically, but they tell you whether there are any outliers (none here) and something about the shape (via the lengths of the whiskers). I'd say the Sect A values are as symmetric as we could hope for. For Sect B, you can say either that they're skewed to the left (and that therefore we have a problem), or that the heights are close enough to symmetric (and that therefore we don't). For me, either is good. As ever, normal quantile plots can offer more insight. With data in this form, the two samples are mixed up, but using facets is the way to go. Philosophically, we draw a normal quantile plot of *all* the heights, and then say at the end that we would actually like a separate plot for each sect:

```
diet %>%
  ggplot(aes(sample = height)) + stat_qq() + stat_qq_line() +
  facet_wrap(~sect, ncol = 1)
```

I decided that I wanted short squat plots rather than tall skinny ones.

With the sizes of the samples, I really don't see any problems here. Most of the evidence for the left skewness in Sect B is actually coming from that largest value being too small. Sect A is as good as you could wish for. Having extreme values being *not extreme enough* is not a problem, since it won't be distorting the mean.

The other way of doing this is to use `filter` to pull out the rows you want and then feed that into the plot:

```
secta <- filter(diet, sect == "a") %>%
  ggplot(aes(sample = sect)) + stat_qq() + stat_qq_line()
```

and the same for sect B. This is the usual `ggplot`-in-pipeline thing where you don't have a named data frame in the `ggplot` because it will use whatever came out of the previous step of the pipeline.

■

(d) Run a t-test to determine whether the mean heights differ significantly. What do you conclude? Explain briefly. (Run the t-test even if your previous work suggests that it is not the right thing to do.)

Solution

The wording states that a two-sided test is correct, which is the default, so you don't need anything special:

```
t.test(height ~ sect, data = diet)
```

```
    Welch Two Sample t-test

data:  height by sect
t = -1.7393, df = 14.629, p-value = 0.103
alternative hypothesis: true difference in means between group a and group b is not equal to
95 percent confidence interval:
 -12.007505   1.229728
sample estimates:
mean in group a mean in group b
       144.8333        150.2222
```

This is a two-sample test, so it takes a `data=`.

Our null hypothesis is that the two sects have equal mean height. The P-value of 0.103 is larger than 0.05, so we do not reject that null hypothesis. That is, there is no evidence that the sects differ in mean height. (That is, our earlier thought that the boys in Sect B were taller is explainable by chance.)

You *must* end up with a statement about mean heights, and when you do a test, you must state the conclusion in the context of the problem, whether I ask you to or not. "Don't reject

the null hypothesis" is a step on the way to an answer, not an answer in itself. If you think it's an answer in itself, you won't be of much use to the world as a statistician.

You might have been thinking that Mood's median test was the thing, if you were worried about that skewness in Sect B. My guess is that the t-test is all right, so it will be the better test (and give the smaller P-value) here, but if you want to do it, you could do it this way:

```
library(smmr)
median_test(diet, height, sect)
```

```
$table
     above
group above below
    a     4     7
    b     6     3

$test
       what     value
1 statistic 1.8181818
2        df 1.0000000
3   P-value 0.1775299
```

My suspicion (that I wrote before doing the test) is correct: there is even less evidence of a difference in median height between the sects. The table shows that both sects are pretty close to 50–50 above and below the overall median, and with sample sizes this small, they are certainly not significantly different from an even split. The small frequencies bring a warning about the chi-squared approximation possibly not working (that `smmr` suppresses). We had one like this elsewhere, but there the result was very significant, and this one is very non-significant. However, the implication is the same: even if the P-value is not very accurate (because the expected frequencies for sect B are both 4.5), the conclusion is unlikely to be wrong because the P-value is so far from 0.05.

■

5.13 Handspans of males and females

Take your right hand, and stretch the fingers out as far as you can. The distance between the tip of your thumb and the tip of your little (pinky) finger is your handspan. The students in a Statistics class at Penn State measured their handspans and also whether they identified as male or female. The data are at http://ritsokiguess.site/datafiles/handspan.txt, with handspans measured in inches. Thinking of these as a random sample of all possible

students, is it true that males have a larger mean handspan than females? This is what we will explore.

(a) Read in and display (some of) the data.

Solution

This is a delimited (by spaces) file, so:

```
library(tidyverse)
my_url <- "http://ritsokiguess.site/datafiles/handspan.txt"
span <- read_delim(my_url, " ")
```

```
Rows: 190 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): sex
dbl (1): handspan

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
span
```

```
# A tibble: 190 x 2
   sex   handspan
   <chr>    <dbl>
 1 M         21.5
 2 M         22.5
 3 M         23.5
 4 F         20
 5 F         19
 6 F         20.5
 7 F         20.5
 8 F         20.2
 9 M         23
10 M         24.5
# i 180 more rows
```

■

(b) Make a suitable graph of the two columns.

Solution

One quantitative variable and one categorical one, so a boxplot:

```
ggplot(span, aes(x=sex, y=handspan)) + geom_boxplot()
```

■

(c) Run a suitable two-sample t-test to address the question of interest. What do you conclude, in the context of the data?

Solution

We are trying to show that males have a *larger* mean handspan, so we need an `alternative`. To see which: there are two sexes, F and M in that order, and we are trying to show that F is less than M:

```
t.test(handspan~sex, data=span, alternative="less")
```

```
    Welch Two Sample t-test

data:  handspan by sex
t = -10.871, df = 187.92, p-value < 2.2e-16
```

```
alternative hypothesis: true difference in means between group F and group M is less than 0
95 percent confidence interval:
      -Inf -2.154173
sample estimates:
mean in group F mean in group M
       20.01699        22.55747
```

The P-value is very small, so there is no doubt that males have larger average handspans than females.

■

(d) Obtain a 90% confidence interval for the difference in mean handspan between males and females. Do you need to run any more code? Explain briefly.

Solution

A confidence interval is two-sided, so we have to re-run the test without the `alternative` to make it two-sided. Note also that we need a 90% interval, which is different from the default 95%, so we have to ask for that too:

```r
t.test(handspan~sex, data=span, conf.level=0.90)
```

```
    Welch Two Sample t-test

data:  handspan by sex
t = -10.871, df = 187.92, p-value < 2.2e-16
alternative hypothesis: true difference in means between group F and group M is not equal to
90 percent confidence interval:
 -2.926789 -2.154173
sample estimates:
mean in group F mean in group M
       20.01699        22.55747
```

The interval is -2.93 to -2.15, *which you should say*. It would be even better to say that males have a mean handspan between 2.15 and 2.93 inches larger than that of females. You also need to round off your answer: the data are given to 0 or 1 decimals, so your interval should be given to 1 or 2 decimals (since the confidence interval is for a mean).

On a question like this, the grader is looking for three things:

- getting the output

- saying what the interval is
- rounding it to a suitable number of decimals.

Thus, getting the output alone is only one out of three things.

■

(e) Explain briefly why you might have some concerns about the validity of the t-tests you ran in this question. Or, if you don't have any concerns, explain briefly why that is.

Solution

The major assumption here is that the male and female handspans have (approximate) normal distributions. The boxplots we drew earlier both had low-end outliers, so the normality is questionable.

Also, say something about the sample sizes and whether or not you think they are large enough to be helpful.

How big are our sample sizes?

```
span %>% count(sex)
```

```
# A tibble: 2 x 2
  sex       n
  <chr> <int>
1 F       103
2 M        87
```

My suspicion is that we are saved by two things: the sample sizes are large enough for the central limit theorem to help us, and in any case, the conclusion is so clear that the assumptions can afford to be off by a bit.

Extra: one way to think about whether we should be concerned about the lack of normality is to use the `bootstrap` to see what the sampling distribution of the sample mean might look like for males and for females. (This is the stuff in Lecture 5a.) The way this works is to sample from each distribution *with replacement*, work out the mean of each sample, then repeat many times, once for the females and once for the males.

To start with the females, the first thing to do is to grab *only* the rows containing the females. This, using an idea from Lecture 5a that we see again properly later, is `filter`:

```
span %>% filter(sex=="F") -> females
females
```

```
# A tibble: 103 x 2
   sex   handspan
   <chr>    <dbl>
 1 F         20
 2 F         19
 3 F         20.5
 4 F         20.5
 5 F         20.2
 6 F         20
 7 F         18
 8 F         20.5
 9 F         22
10 F         20
# i 93 more rows
```

There are 103 females. From these we need to take a "large" number of bootstrap samples to get a sense of how the mean handspan of the females varies:

```
set.seed(457299)
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(the_sample = list(sample(females$handspan, replace = TRUE))) %>%
  mutate(the_mean = mean(the_sample)) -> d
```

Then we make a histogram of the bootstrap sampling distribution of the sample mean for the females:

```
ggplot(d, aes(x = the_mean)) + geom_histogram(bins = 10)
```

I don't know what you think of this. There are a few more extreme values than I would like, and it looks otherwise a bit left-skewed to me. But maybe I am worrying too much.

The males one works exactly the same way:

```
span %>% filter(sex=="M") -> males
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(the_sample = list(sample(males$handspan, replace = TRUE))) %>%
  mutate(the_mean = mean(the_sample)) -> d
```

```
ggplot(d, aes(x = the_mean)) + geom_histogram(bins = 10)
```

There is a similar story here. I think these are good enough overall, and so I am happy with the two-sample t-test, but it is not as clear-cut as I was expecting.

■

5.14 The anchoring effect: Australia vs US

Two groups of students (in a class at a American university) were asked what they thought the population of Canada was. (The correct answer at the time was just over 30 million.) Some of the students, before having to answer this, were told that the population of the United States was about 270 million. The other students in the class were told that the population of Australia was about 18 million. The data are in http://ritsokiguess.site/datafiles/anchoring.csv. The first column contains the country whose population the student was told, and the second contains the student's guess at the population of Canada.

You might wonder how being told the population of an unrelated country would have any impact on a student's guess at the population of Canada. Psychology says it does: it's called the *anchoring effect*, and the idea is that the number mentioned first acts as an "anchor": a person's guess will be closer to the anchor than it would have been otherwise. In this case, that would mean that the guesses for the students given the US as an anchor will be higher than for the students given Australia as an anchor. We are interested in seeing whether there is evidence for that here.

(a) Read in and display (some of) the data.

Solution

I made it as easy as I could:

```
my_url <- "http://ritsokiguess.site/datafiles/anchoring.csv"
canada <- read_csv(my_url)
```

```
Rows: 21 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): anchor
dbl (1): estimate

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
canada
```

```
# A tibble: 21 x 2
   anchor estimate
   <chr>     <dbl>
 1 US         20
 2 US         90
 3 US          1.5
 4 US        100
 5 US        132
 6 US        150
 7 US        130
 8 US         40
 9 US        200
10 US         20
# i 11 more rows
```

You might need to scroll down to see that both "anchor" countries are indeed represented.

■

(b) Draw a suitable graph of these data.

Solution

One categorical variable and one quantitative one, so a boxplot:

```
ggplot(canada, aes(x = anchor, y = estimate)) + geom_boxplot()
```

■

(c) Explain briefly why a Welch t-test would be better than a pooled t-test in this case.

Solution

The decision between these two tests lies in whether you think the two groups have equal spread (variance, strictly). Here, the spread for the US group is much larger than for the Australia group, even taking into account the big outlier in the latter group. Since the spreads are different, we should do a Welch t-test rather than a pooled one.

Make sure you answer the question I asked, not the one you think I should have asked.

There is a separate question about whether the groups are close enough to normal, but I wasn't asking about that here. I was asking: *given* that we have decided to do some kind of t-test, why is the Welch one better than the pooled one? I am not asking whether we should be doing any kind of t-test at all; if I had, you could *then* reasonably talk about the outlier in the Australia group, and other possible skewness in its distribution, but that's not what I asked about.

■

(d) Run a suitable Welch t-test and display the output.

Solution

The word "suitable" is a hint that you may have to think a bit about how you run the test. If the anchoring effect is real, the mean of the guesses for the students told the population of the US will be higher on average than for those told the population of Australia, so we want a one-sided alternative. Australia is before the US alphabetically, so the alternative has to be `less`:

```
t.test(estimate~anchor, data = canada, alternative = "less")
```

```
    Welch Two Sample t-test

data:  estimate by anchor
t = -3.0261, df = 10.558, p-value = 0.006019
alternative hypothesis: true difference in means between group australia and group US is less
95 percent confidence interval:
      -Inf -26.63839
sample estimates:
mean in group australia        mean in group US
               22.45455                88.35000
```

Note that the Welch test is the default, so you don't have to do anything special to get it. Your output will tell you that a Welch test is what you have. It's if you want a *pooled* test that you have to ask for it specifically (with `var.equal = TRUE`).

If you get a P-value close to 1, this is often an indication that you have the alternative the wrong way around.

■

(e) What do you conclude from your test, in the context of the data?

Solution

The P-value is definitely less than 0.05, so we reject the null hypothesis (which says that the mean guess is the same regardless of the anchor the student was given). So we have evidence that the mean guess is higher for the students who were given the US population first.

Extra 1: this is perhaps the place to think about what effect that outlier in the `australia` group might have had. Since it is a high outlier, its effect will be to make the the `australia`

mean higher than it would have been otherwise, and therefore to make the two group means closer together. Despite this, the difference still came out strongly significant, so that we can be *even more sure than the P-value says* that there is a real difference between the means of estimates of the population of Canada. (To say it differently, if the outlier had not been there, the difference in means would have been even bigger and thus even more significant.)

Extra 2: if you are still worried about doing a two-sample t-test here, you might consider looking at the bootstrapped sampling distribution of the sample mean of the `australia` group:

```
canada %>% filter(anchor == "australia") -> oz
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(the_sample = list(sample(oz$estimate, replace = TRUE))) %>%
  mutate(the_mean = mean(the_sample)) %>%
  ggplot(aes(x = the_mean)) + geom_histogram(bins=10)
```

This is indeed skewed to the right (though, with 11 observations, not nearly so non-normal as the original data), and so the P-value we got from the t-test may not be reliable. But, as discussed in Extra 1, the "correct" P-value is, if anything, *even smaller* than the one we got, and so the conclusion we drew earlier (that there is a significant anchoring effect) is not going to change.

Extra 3: looking even further ahead, there is a test that definitely *does* apply here, called Mood's Median Test. You won't have installed the package yet, so this won't work for you

just yet (read ahead if you want to learn more), but here's how it goes:

```
library(smmr)
median_test(canada, estimate, anchor)
```

```
$table
           above
group       above below
  australia     2     5
  US            7     1

$test
       what      value
1 statistic 5.40178571
2        df 1.00000000
3   P-value 0.02011616
```

This does (as it is written) a two-sided test, because it can also be used for comparing more than two groups. Since we want a one-sided test here, you can (i) check that we are on the correct side (we are)[12] (ii) halve the P-value to get 0.010.

This is a P-value you can trust. It is not smaller than the t-test one, perhaps because this test is less powerful than the t-test in most cases.[13]

■

[12]The test works by comparing the data values in each group to the overall median. The students who were given Australia as an anchor mostly guessed below the overall median, and the students given the US as an anchor mostly guessed above.

[13]It uses the data less efficiently than the t-test; it just counts the number of values above and below the overall median in each group, rather than using the actual numbers to compute means.

6 Power and sample size

```
library(tidyverse)
```

6.1 Simulating power

This question investigates power by simulation.

(a) Use `rnorm` to generate 10 random values from a normal distribution with mean 20 and SD 2. Do your values look reasonable? Explain briefly. (You don't need to draw a graph.)

(b) Estimate by simulation the power of a t-test to reject a null hypothesis of 20 when the true mean is also 20, the population SD is 2, and the sample size is 10, against a (default) two-sided alternative. Remember the steps: (i) generate a lot of random samples from the true distribution, (ii) run the t-test with the required null mean, (iii) pull out the P-values, (iv) count how many of them are 0.05 or less.

(c) In the simulation you just did, was the null hypothesis true or false? Do you want to reject the null hypothesis or not? Explain briefly why the simulation results you got were (or were not) about what you would expect.

(d) By copying, pasting and editing your code from the previous part, estimate the power of the test of $H_0 : \mu = 20$ (against a two-sided alternative) when the true population mean is 22 (rather than 20).

(e) Use R to calculate this power exactly (without simulation). Compare the exact result with your simulation.

6.2 Calculating power and sample size for estimating mean

We are planning a study to estimate a population mean. The population standard deviation is believed to be 20, and the population distribution is believed to be approximately normal. We will be testing the null hypothesis that the population mean is 100. Suppose the population

mean is actually 110, and we want to determine how likely we are to (correctly) reject the null hypothesis in this case, using a two-sided (but one-sample) test with $\alpha = 0.05$.

(a) We will take a sample of size $n = 30$. Calculate the power of this test.

(b) Find the sample size necessary to obtain a power of at least 0.80 under these conditions. What sample size do you need? Explain briefly how your answer is consistent with (a).

6.3 Simulating power for proportions

In opinion surveys (and other places), we are testing for a proportion p (for example, the proportion of people agreeing with some statement). Often, we want to know whether the proportion is "really" greater than 0.5.[1]
That would entail testing a null $H_0 : p = 0.5$ against an alternative $H_a : p > 0.5$. This is usually done by calculating the test statistic

$$z = \frac{\hat{p} - 0.5}{\sqrt{0.25/n}},$$

where $\hat{p}$ is the observed proportion in the sample, and getting a P-value from the upper tail of a standard normal distribution. (The 0.25 is $p(1-p)$ where $p = 0.5$.) This is what `prop.test` does, as we investigate shortly.

(a) Use `rbinom` to generate a random value from a binomial distribution with $n = 100$ and $p = 0.6$. There are three inputs to `rbinom`: the first one should be the number 1, and the second and third are the n and p of the binomial distribution.

(b) Using the random binomial that you generated just above, use `prop.test` to test whether it could reasonably have come from a binomial population with $n = 100$ and $p = 0.5$, or whether p is actually bigger than 0.5. (Of course, you know it actually did not come from a population with $p = 0.5$.) `prop.test` has, for us, four inputs, thus:

- the observed number of successes
- the `n` of the binomial distribution
- the null-hypothesis `p` of the binomial distribution
- the alternative hypothesis, here "greater"

(c) Run `prop.test` again, just as you did before, but this time save the result, and extract the piece of it called `p.value`. Is that the P-value from your test?

[1] That would mean assessing whether an observed proportion could be greater than 0.5 just by chance, or whether it is bigger enough than 0.5 to reject chance as a plausible explanation.

(d) Estimate the power of a test of $H_0 : p = 0.5$ against $H_a : p > 0.5$ when $n = 500$ and $p = 0.56$, using $\alpha = 0.05$. There are three steps:

- generate random samples from binomial distributions with $n = 500$ and $p = 0.56$, repeated "many" times (something like 1000 or 10,000 is good)
- run `prop.test` on each of those random samples
- extract the P-value for each test and save the results (in a column called, perhaps, `pvals`).

So I lied: the fourth and final step is to count how many of those P-values are 0.05 or less.

6.4 Designing a study to have enough power

You are designing a study to test the null hypothesis that a population mean is 0 against the alternative hypothesis that it is greater than 0. Assume that the population SD is $\sigma = 15$. It is important to detect the alternative $\mu = 2$; that is, we want to design the study so that most of the time the null hypothesis would be (correctly) rejected if in fact $\mu = 2$. A one-sample t-test will be used, and the data values are assumed to have a normal distribution.

(a) Use simulation to estimate the power of this test when the sample size is 100. Use $\alpha = 0.05$.

(b) Again by using simulation, estimate how large a sample size would be needed to obtain a power of 0.80. Show and briefly explain your process.

6.5 Power and α in a skewed population

A population of a large number of values `v` is at http://ritsokiguess.site/datafiles/pop.csv, in a CSV file.

(a) Read in the population and display some of the values.

(b) Obtain a suitable plot of your population. What do you notice?

(c) If you take a sample of 10 observations from this population and run a t-test, how likely are you to (correctly) reject the null hypothesis $H_0 : \mu = 4$, against the alternative $H_a : \mu > 4$? Investigate by simulation.

(d) Try again with a sample size of 50 (leaving everything else the same). Explain briefly why the results so far are as you'd expect.

(e) Again by simulation, estimate the probability that the null hypothesis $H_0 : \mu = 5$ will be rejected when a sample of size 10 is taken from this population, in favour of the alternative $H_a : \mu > 5$. Explain briefly why the answer is not what you would have expected, and why that happened here. (Hint: what is the population mean?)

My solutions follow:

6.6 Simulating power

This question investigates power by simulation.

(a) Use `rnorm` to generate 10 random values from a normal distribution with mean 20 and SD 2. Do your values look reasonable? Explain briefly. (You don't need to draw a graph.)

Solution

`rnorm` with the number of values first, then the mean, then the SD:

```
x=rnorm(10,20,2)
x
```

```
[1] 21.59476 18.64044 21.83231 18.76556 18.64861 21.81889 21.62614 20.18249
[9] 16.91266 20.63490
```

95% of the sampled values should be within 2 SDs of the mean, that is, between 16 and 24 (or 99.7% should be within 3 SDs of the mean, between 14 and 26). None of my values are even outside the interval 16 to 24, though yours may be different.

I saved mine in a variable and then displayed them, which you don't need to do. I did because there's another way of assessing them for reasonableness: turn the sample into z-scores and see whether the values you get look like z-scores (that is, most of them are between -2 and 2, for example):

```
(x-20)/2
```

```
[1]  0.79738130 -0.67977910  0.91615386 -0.61722168 -0.67569291  0.90944266
[7]  0.81307163  0.09124563 -1.54367207  0.31744905
```

These ones look very much like z-scores. This, if you think about it, is really the flip-side of 68–95–99.7, so it's another way of implementing the same idea.

You might also think of finding the *sample* mean and SD, and demonstrating that they are close to the right answers. Mine are:

```
mean(x)
```

```
[1] 20.06568
```

```
sd(x)
```

```
[1] 1.731305
```

The sample SD is more variable than the sample mean, so it can get further away from the population SD than the sample mean does from the population mean.

The downside to this idea is that it doesn't get at assessing the normality, which looking at z-scores or equivalent does. Maybe coupling the above with a boxplot would have helped, had I not said "no graphs", since then you'd (hopefully) see no outliers and a roughly symmetric shape.

This is old-fashioned "base R" technology; you could do it with a data frame like this:

```
d <- tibble(x=rnorm(10,20,2))
d
```

```
# A tibble: 10 x 1
       x
   <dbl>
 1  20.6
 2  19.3
 3  20.2
 4  18.7
 5  21.3
 6  17.8
 7  19.8
 8  20.9
 9  21.7
10  20.6
```

```
d %>% summarize(m=mean(x), s=sd(x))
```

```
# A tibble: 1 x 2
      m     s
  <dbl> <dbl>
1  20.1  1.21
```

These are different random numbers, but are about equally what you'd expect. (These ones are a bit less variable than you'd expect, but with only ten values, don't expect perfection.)

Some discussion about the kind of values you should get, and whether or not you get them, is what is called for here. I want you to say something convincing about how the values you get come from a normal distribution with mean 20 *and* SD 2. "Close to 20" is not the whole answer here, because that doesn't get at "how close to 20?": that is, it talks about the mean but not about the SD.

■

(b) Estimate by simulation the power of a t-test to reject a null hypothesis of 20 when the true mean is also 20, the population SD is 2, and the sample size is 10, against a (default) two-sided alternative. Remember the steps: (i) generate a lot of random samples from the true distribution, (ii) run the t-test with the required null mean, (iii) pull out the P-values, (iv) count how many of them are 0.05 or less.

Solution

Once you get the hang of these, they all look almost the same. This one is easier than some because we don't have to do anything special to get a two-sided alternative hypothesis. The initial setup is to make a dataframe with a column called something like `sim` to label the simulations, and then a `rowwise` to generate one random sample, t-test and P-value for each simulation:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(norm_sample = list(rnorm(10, 20, 2))) %>%
  mutate(t_test = list(t.test(norm_sample, mu = 20))) %>%
  mutate(pval = t_test$p.value) %>%
  count(pval <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `pval <= 0.05`     n
```

```
  <lgl>          <int>
1 FALSE            958
2 TRUE              42
```

The power is about 4.2%. This seems depressingly small, but see the next part. (Are you confused about something in this one? You have a right to be.)

■

(c) In the simulation you just did, was the null hypothesis true or false? Do you want to reject the null hypothesis or not? Explain briefly why the simulation results you got were (or were not) about what you would expect.

Solution

The null mean and the true mean were both 20: that is, the null hypothesis was correct, and rejecting it would be a mistake, to be precise a type I error. We were doing the test at $\alpha = 0.05$ (by comparing our collection of simulated P-values with 0.05), so we should be making a type I error 5% of the time. This is entirely in line with the 4.2% of (wrong) rejections that I had. Your estimation is likely to be different from mine, but you should be rejecting about 5% of the time. If your result is very different from 5%, that's an invitation to go back and check your code. On the other hand, if it *is* about 5%, that ought to give you confidence to go on and use the same ideas for the next part.

■

(d) By copying, pasting and editing your code from the previous part, estimate the power of the test of $H_0 : \mu = 20$ (against a two-sided alternative) when the true population mean is 22 (rather than 20).

Solution

Here's the code we just used:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(norm_sample = list(rnorm(10, 20, 2))) %>%
  mutate(t_test = list(t.test(norm_sample, mu = 20))) %>%
  mutate(pval = t_test$p.value) %>%
  count(pval <= 0.05)
```

One of those 20s needs to become 22. Not the one in the `t.test`, since the hypotheses have not changed. So we need to change the 20 in the `rnorm` line to 22, since that's where we're generating data from the true distribution. The rest of it stays the same:

```r
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(norm_sample = list(rnorm(10, 20, 2))) %>%
  mutate(t_test = list(t.test(norm_sample, mu = 22))) %>%
  mutate(pval = t_test$p.value) %>%
  count(pval <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `pval <= 0.05`     n
  <lgl>          <int>
1 FALSE            192
2 TRUE             808
```

This time, we *want* to reject, since the null hypothesis is false. So look at the `TRUE` count: the power is about 80%. We are very likely to correctly reject a null of 20 when the mean is actually 22.

Extra: another way to reason that the power should be fairly large is to think about what kind of sample you are likely to get from the true distribution: one with a mean around 22 and an SD around 2. Thus the t-statistic should be somewhere around this (we have a sample size of 10):

```r
t_stat=(22-20)/(2/sqrt(10))
t_stat
```

```
[1] 3.162278
```

and the two-sided P-value should be about

```r
2*(1-pt(t_stat,10-1))
```

```
[1] 0.01150799
```

Of course, with your actual data, you will sometimes be less lucky than this (a sample mean nearer 20 or a larger sample SD), but sometimes you will be luckier. But the suggestion is that most of the time, the P-value will be pretty small and you will end up correctly rejecting.

The quantity `t_stat` above, 3.16, is known to some people as an "effect size", and summarizes how far apart the null and true means are, relative to the amount of variability present (in the sampling distribution of the sample mean). As effect sizes go, this one is pretty large.

■

(e) Use R to calculate this power exactly (without simulation). Compare the exact result with your simulation.

Solution

This is `power.t.test`. The quantity `delta` is the difference between true and null means:

```
power.t.test(n=10,delta=22-20,sd=2,type="one.sample",alternative="two.sided")
```

```
     One-sample t test power calculation

              n = 10
          delta = 2
             sd = 2
      sig.level = 0.05
          power = 0.8030962
    alternative = two.sided
```

This, 0.803, is very close to the value I got from my simulation. Which makes me think I did them both right. This is not a watertight proof, though: for example, I might have made a mistake and gotten lucky somewhere. But it does at least give me confidence.

Extra: when you estimate power by simulation, what you are doing is rejecting or not with a certain probability (which is the same for all simulations). So the number of times you actually *do* reject has a binomial distribution with n equal to the number of simulated P-values you got (1000 in my case; you could do more) and a p that the simulation is trying to estimate. This is inference for a proportion, exactly what `prop.test` does.

Recall that `prop.test` has as input:

- a number of "successes" (rejections of the null in our case)
- the number of trials (simulated tests)
- the null-hypothesis value of `p` (optional if you only want a CI)
- (optional) a confidence level `conf.level`.

In part (b), we knew that the probability of (incorrectly) rejecting should have been 0.05 and we rejected 42 times out of 1000:

```
prop.test(42,1000,0.05)
```

```
    1-sample proportions test with continuity correction

data:  42 out of 1000, null probability 0.05
X-squared = 1.1842, df = 1, p-value = 0.2765
alternative hypothesis: true p is not equal to 0.05
95 percent confidence interval:
 0.03079269 0.05685194
sample estimates:
    p
0.042
```

Looking at the P-value, we definitely fail to reject that the probability of (incorrectly) rejecting is the 0.05 that it should be. Ouch. That's true, but unnecessarily confusing. Look at the confidence interval instead, 0.031 to 0.057. The right answer is 0.05, which is inside that interval, so good.

In part (c), we didn't know what the power was going to be (not until we calculated it with `power.t.test`, anyway), so we go straight for a confidence interval; the default 95% confidence level is fine. We (correctly) rejected 798 times out of 1000:

```r
prop.test(798,1000)
```

```
    1-sample proportions test with continuity correction

data:  798 out of 1000, null probability 0.5
X-squared = 354.02, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.7714759 0.8221976
sample estimates:
    p
0.798
```

I left out the 3rd input since we're not doing a test, and ignore the P-value that comes out. (The default null proportion is 0.5, which often makes sense, but not here.)

According to the confidence interval, the estimated power is between 0.771 and 0.822. This interval definitely includes what we now know is the right answer of 0.803.

This might be an accurate enough assessment of the power for you, but if not, you can do more simulations, say 10,000:

```r
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(norm_sample = list(rnorm(10, 20, 2))) %>%
  mutate(t_test = list(t.test(norm_sample, mu = 22))) %>%
  mutate(pval = t_test$p.value) %>%
  count(pval <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `pval <= 0.05`     n
  <lgl>          <int>
1 FALSE           1923
2 TRUE            8077
```

I copied and pasted my code again, which means that I'm dangerously close to turning it into a function, but anyway.

The confidence interval for the power is then

```r
prop.test(7996,10000)
```

```
    1-sample proportions test with continuity correction

data:  7996 out of 10000, null probability 0.5
X-squared = 3589.2, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.7915892 0.8073793
sample estimates:
     p
0.7996
```

that is, from 0.792 to 0.807, which once again includes the right answer of 0.803. The first interval, based on 1,000 simulations, has length 0.051, while this interval has length 0.015. The first interval is more than three times as long as the second, which is about what you'd expect since the first one is based on 10 times fewer simulations, and thus ought to be a factor of $\sqrt{10} \simeq 3.16$ times longer.

This means that you can estimate power as accurately as you like by doing a large enough (possibly very large) number of simulations. Provided, that is, that you are prepared to wait a possibly long time for it to finish working!

■

6.7 Calculating power and sample size for estimating mean

We are planning a study to estimate a population mean. The population standard deviation is believed to be 20, and the population distribution is believed to be approximately normal. We will be testing the null hypothesis that the population mean is 100. Suppose the population mean is actually 110, and we want to determine how likely we are to (correctly) reject the null hypothesis in this case, using a two-sided (but one-sample) test with $\alpha = 0.05$.

(a) We will take a sample of size $n = 30$. Calculate the power of this test.

Solution

`power.t.test`. Fill in: sample size `n`, difference in means `delta` ($10 = 110 - 100$), population SD `sd`, type of test `type` (`one.sample`) and kind of alternative hypothesis `alternative` (`two.sided`). Leave out `power` since that's what we want:

```
power.t.test(n=30,delta=10,sd=20,type="one.sample",alternative="two.sided")
```

```
     One-sample t test power calculation

              n = 30
          delta = 10
             sd = 20
      sig.level = 0.05
          power = 0.7539627
    alternative = two.sided
```

I meant "calculate" exactly rather than "estimate" (by simulation). Though if you want to, you can do that as well, thus:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(samples = list(rnorm(30, 110, 20))) %>%
  mutate(ttest = list(t.test(samples, mu= 100))) %>%
  mutate(pvals = ttest$p.value) %>%
  count(pvals<=0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `pvals <= 0.05`     n
  <lgl>           <int>
1 FALSE             257
2 TRUE              743
```

That came out alarmingly close to the exact answer.

■

(b) Find the sample size necessary to obtain a power of at least 0.80 under these conditions. What sample size do you need? Explain briefly how your answer is consistent with (a).

Solution

Again, the implication is "by calculation". This time, in `power.t.test`, put in 0.80 for `power` and leave out `n`. The order of things doesn't matter (since I have named everything that's going into `power.t.test`):

```
power.t.test(delta=10,power=0.80,sd=20,type="one.sample",alternative="two.sided")
```

```
     One-sample t test power calculation

              n = 33.3672
          delta = 10
             sd = 20
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

To get sample size for power at least 0.80, we have to round 33.36 *up* to the next whole number, ie. $n = 34$ is needed. (A sample of size 33 wouldn't quite have enough power.)

This answer is consistent with (a) because a sample size of 30 gave a power a bit less than 0.80, and so to increase the power by a little (0.75 to 0.80), we had to increase the sample size by a little (30 to 34).

Extra: estimating sample sizes by simulation is tricky, because the sample size has to be input to the simulation. That means your only strategy is to try different sample sizes until you find one that gives the right power.

In this case, we know that a sample of size 30 doesn't give quite enough power, so we have to up the sample size a bit. How about we try 40? I copied and pasted my code from above and changed 30 to 40:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(samples = list(rnorm(40, 110, 20))) %>%
  mutate(ttest = list(t.test(samples, mu= 100))) %>%
  mutate(pvals = ttest$p.value) %>%
  count(pvals<=0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `pvals <= 0.05`     n
  <lgl>           <int>
1 FALSE             130
2 TRUE              870
```

Now the power is a bit too big, so we don't need a sample size quite as big as 40. So probably our next guess would be 35. But before we copy and paste again, we should be thinking about making a function of it first, with the sample size as input. Copy-paste once more and edit:

```
sim_power=function(n) {
  tibble(sim = 1:1000) %>%
    rowwise() %>%
    mutate(samples = list(rnorm(n, 110, 20))) %>%
    mutate(ttest = list(t.test(samples, mu= 100))) %>%
    mutate(pvals = ttest$p.value) %>%
    ungroup() %>%
    count(pvals<=0.05)
}
```

In the grand scheme of things, we might want to have the null and true means, population SD and α be inputs to the function as well, so that we have a more general tool, but this will do for now.

Let's run it with a sample size of 35:

```
sim_power(35)
```

```
# A tibble: 2 x 2
```

```
  `pvals <= 0.05`     n
  <lgl>           <int>
1 FALSE             183
2 TRUE              817
```

and I'm going to call that good. (Because there is randomness in the estimation of the power, don't expect to get *too* close to the right answer. This one came out a fair bit less than the right answer; the power for $n = 35$ should be a bit *more* than 0.80.)

Now that you have the software to do it, you can see that figuring out a sample size like this, at least roughly, won't take very long: each one of these simulations takes maybe seconds to run, and all you have to do is copy and paste the previous one, and edit it to contain the new sample size before running it again. You're making the computer work hard while you lazily sip your coffee, but there's no harm in that: programmer's brain cells are more valuable than computer CPU cycles, and you might as well save your brain cells for when you really need them.

You might even think about automating this further. The easiest way, now that we have the function, is something like this:

```r
tibble(ns = seq(20, 50, 5)) %>%
  rowwise() %>%
  mutate(power_tab = list(sim_power(ns))) %>%
  unnest(power_tab) %>%
  pivot_wider(names_from = `pvals <= 0.05`, values_from = n)
```

```
# A tibble: 7 x 3
     ns `FALSE` `TRUE`
  <dbl>   <int>  <int>
1    20     402    598
2    25     321    679
3    30     258    742
4    35     188    812
5    40     132    868
6    45     100    900
7    50      89    911
```

The business end of this is happening in the first three lines. I wasn't thinking of this when I originally wrote `sim_power` to return a dataframe, so there is a bit more fiddling after the simulations are done: I have to `unnest` to see what the list-column `power_tab` actually contains, and because of the layout of the output from unnesting `sim_power` (long format), it

looks better if I pivot it wider, so that I can just cast my eye down the TRUE column and see the power increasing as the sample size increases.

You might also think of something like bisection to find the sample size that has power 0.8, but it starts getting tricky because of the randomness; just by chance, it may be that sometimes the simulated power goes *down* as the sample size goes up. With 1000 simulations each time, it seems that the power ought to hit 80% with a sample size between 30 and 35.

6.8 Simulating power for proportions

In opinion surveys (and other places), we are testing for a proportion p (for example, the proportion of people agreeing with some statement). Often, we want to know whether the proportion is "really" greater than 0.5.[2]
That would entail testing a null $H_0 : p = 0.5$ against an alternative $H_a : p > 0.5$. This is usually done by calculating the test statistic

$$z = \frac{\hat{p} - 0.5}{\sqrt{0.25/n}},$$

where $\hat{p}$ is the observed proportion in the sample, and getting a P-value from the upper tail of a standard normal distribution. (The 0.25 is $p(1-p)$ where $p = 0.5$.) This is what `prop.test` does, as we investigate shortly.

(a) Use `rbinom` to generate a random value from a binomial distribution with $n = 100$ and $p = 0.6$. There are three inputs to `rbinom`: the first one should be the number 1, and the second and third are the n and p of the binomial distribution.

Solution

I am doing some preparatory work that you don't need to do:

```
set.seed(457299)
```

By setting the "seed" for the random number generator, I guarantee that I will get the same answers every time I run my code below (and therefore I can talk about my answers without worrying that they will change). Up to you whether you do this. You can "seed" the random number generator with any number you like. A lot of people use `1`. Mahinda seems to like `123`. Mine is an old phone number.

And so to work:

```
rbinom(1, 100, 0.6)
```

[2]That would mean assessing whether an observed proportion could be greater than 0.5 just by chance, or whether it is bigger enough than 0.5 to reject chance as a plausible explanation.

```
[1] 60
```

I got exactly 60% successes this time. You probably won't get exactly 60, but you should get somewhere close. (If you use my random number seed and use the random number generator exactly the same way I did, you should get the same values I did.)

For fun, you can see what happens if you change the 1:

```
rbinom(3, 100, 0.6)
```

```
[1] 58 57 55
```

Three random binomials, that happened to come out just below 60. We're going to leave the first input as 1, though, and let `rowwise` handle "lots of sampled values" later.

■

(b) Using the random binomial that you generated just above, use `prop.test` to test whether it could reasonably have come from a binomial population with $n = 100$ and $p = 0.5$, or whether p is actually bigger than 0.5. (Of course, you know it actually did not come from a population with $p = 0.5$.) `prop.test` has, for us, four inputs, thus:

- the observed number of successes
- the `n` of the binomial distribution
- the null-hypothesis `p` of the binomial distribution
- the alternative hypothesis, here "greater"

Solution

I got exactly 60 successes, so I do this:

```
prop.test(60, 100, 0.5, alternative = "greater")
```

```
	1-sample proportions test with continuity correction

data:  60 out of 100, null probability 0.5
X-squared = 3.61, df = 1, p-value = 0.02872
alternative hypothesis: true p is greater than 0.5
95 percent confidence interval:
 0.5127842 1.0000000
```

```
sample estimates:
  p
0.6
```

The P-value should at least be fairly small, since 60 is a bit bigger than 50. (Think about tossing a coin 100 times; would 60 heads make you doubt the coin's fairness? The above says it should.)

■

(c) Run `prop.test` again, just as you did before, but this time save the result, and extract the piece of it called `p.value`. Is that the P-value from your test?

Solution

Copying and pasting:

```
p_test <- prop.test(60, 100, 0.5, alternative = "greater")
p_test$p.value
```

```
[1] 0.02871656
```

Yep, the same.

■

(d) Estimate the power of a test of $H_0 : p = 0.5$ against $H_a : p > 0.5$ when $n = 500$ and $p = 0.56$, using $\alpha = 0.05$. There are three steps:

- generate random samples from binomial distributions with $n = 500$ and $p = 0.56$, repeated "many" times (something like 1000 or 10,000 is good)
- run `prop.test` on each of those random samples
- extract the P-value for each test and save the results (in a column called, perhaps, `pvals`).

So I lied: the fourth and final step is to count how many of those P-values are 0.05 or less.

Solution

The first part of the first step is to create a column called something like `sim` that labels each simulated sample, and to make sure that everything happens rowwise. After that, you follow the procedure:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
```

```r
  mutate(sample = rbinom(1, 500, 0.56)) %>%
  mutate(test = list(prop.test(sample, 500, 0.5, alternative = "greater"))) %>%
  mutate(pvals = test$p.value) %>%
  count(pvals <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `pvals <= 0.05`     n
  <lgl>           <int>
1 FALSE             143
2 TRUE              857
```

The previous parts, using `rbinom` and `prop.test`, were meant to provide you with the ingredients for this part. The first step is to use `rbinom`. The first input is 1 since we only want one random binomial each time (the `rowwise` will handle the fact that you actually want lots of them; you only want one *per row* since you are working rowwise). The second step runs `prop.test`; the first input to that is each one of the numbers of successes from the first step. The last part is to pull out all the P-values and make a table of them, just like the example in lecture.

The estimated power is about 85%. That is, if p is actually 0.56 and we have a sample of size 500, we have a good chance of (correctly) rejecting that $p = 0.5$.

Extra: It turns out that SAS can work out this power by calculation (using `proc power`). SAS says our power is also about 85%, as our simulation said. I was actually pleased that my simulation came out so close to the right answer.

In contrast to `power.t.test`, SAS's `proc power` handles power analyses for a lot of things, including analysis of variance, correlation and (multiple) regression. What these have in common is some normal-based theory that allows you (under assumptions of sufficiently normal-shaped populations) to calculate the exact answer (that is, the distribution of the test statistic when the *alternative* hypothesis is true). The case we looked at is one of those because of the normal approximation to the binomial: once n gets big, particularly if p is somewhere near 0.5, the binomial is very well approximated by a normal with the right mean and SD.

The moral of this story is that when you have a decently large sample, $n = 500$ in this case, p doesn't have to get very far away from 0.5 before you can correctly reject 0.5. Bear in mind that sample sizes for estimating proportions need to be larger than those for estimating means, so $n = 500$ is large without being huge. The practical upshot is that if you design a survey and give it to 500 (or more) randomly chosen people, the proportion of people in favour doesn't have to be much above 50% for you to correctly infer that it *is* above 50%, most of the time.

■

6.9 Designing a study to have enough power

You are designing a study to test the null hypothesis that a population mean is 0 against the alternative hypothesis that it is greater than 0. Assume that the population SD is $\sigma = 15$. It is important to detect the alternative $\mu = 2$; that is, we want to design the study so that most of the time the null hypothesis would be (correctly) rejected if in fact $\mu = 2$. A one-sample t-test will be used, and the data values are assumed to have a normal distribution.

(a) Use simulation to estimate the power of this test when the sample size is 100. Use $\alpha = 0.05$.

Solution

Use at least 1000 simulations (more, if you're willing to wait for it). In `rnorm`, the sample size is first, then the (true) population mean, then the (assumed) population SD:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(100, 2, 15))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 0, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE               639
2 TRUE                361
```

The power is (estimated as) a disappointing 0.361. Your answer won't (most likely) be the same as this, but it should be somewhere close. I would like to see you demonstrate that you know what power is, for example "if the population mean is actually 2, the null hypothesis $H_0 : \mu = 0$, which is wrong, will only be rejected about 36% of the time".[3]

The test we are doing is one-sided, so you need the `alternative` in there. If you omit it, you'll have the answer to a different problem:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
```

[3]This is why I called my result disappointing. I would like to reject a lot more of the time than this, but, given that the truth was not very far away from the null given the (large) population SD, I can't. See Extra 1.

```
  mutate(my_sample = list(rnorm(100, 2, 15))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 0))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE               740
2 TRUE                260
```

This is the probability that you reject $H_0 : \mu = 0$ in favour of $H_a : \mu \neq 0$. This is smaller, because the test is "wasting effort" allowing the possibility of rejecting when the sample mean is far enough *less* than zero, when most of the time the samples drawn from the true distribution have mean greater than zero. (If you get a sample mean of 2.5, say, the P-value for a one-sided test will be smaller than for a two-sided one.)

Extra 1:

This low power of 0.361 is because the population SD is large relative to the kind of difference from the null that we are hoping to find. To get a sense of how big the power might be, imagine you draw a "typical" sample from the true population: it will have a sample mean of 2 and a sample SD of 15, so that t will be about

```
(2-0)/(15/sqrt(100))
```

```
[1] 1.333333
```

You won't reject with this (t would have to be bigger than 2), so in the cases where you *do* reject, you'll have to be more lucky: you'll need a sample mean bigger than 2, or a sample SD smaller than 15. So the power won't be very big, less than 0.5, because about half the time you'll get a test statistic less than 1.33 and about half the time more, and not all of *those* will lead to rejection.

Extra 2:

This is exactly the situation where `power.t.test` works, so we can get the exact answer (you need all the pieces):

```
power.t.test(n=100, delta=2-0, sd=15, type="one.sample",
alternative = "one.sided")
```

```
     One-sample t test power calculation

              n = 100
          delta = 2
             sd = 15
      sig.level = 0.05
          power = 0.3742438
    alternative = one.sided
```

Your answer, from 1000 simulations, should be within about 3 percentage points of that. (Mine was only about 1 percentage point off.)

■

(b) Again by using simulation, estimate how large a sample size would be needed to obtain a power of 0.80. Show and briefly explain your process.

Solution

The point of this one is the process as well as the final answer, so you need to show and justify what you are doing. Showing only a final answer *does not* show that you know how to do it. The *whole point* of this one is to make mistakes and fix them!

The simulation approach does not immediately give you a sample size for fixed power, so what you have to do is to try different sample sizes until you get one that gives a power close enough to 0.80. You have to decide what "close enough" means for you, given that the simulations have randomness in them. I'm going to use 10,000 simulations for each of my attempts, in the hope of getting a more accurate answer.

First off, for a sample size of 100, the power was too small, so the answer had better be bigger than 100. I'll try 200. For these, copy and paste the code, changing the sample size each time:

```r
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(200, 2, 15))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 0, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
```

```
  <lgl>             <int>
1 FALSE              4064
2 TRUE               5936
```

A sample size of 200 isn't big enough yet. I'll double again to 400:

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(400, 2, 15))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 0, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE              1582
2 TRUE               8418
```

Getting closer. 400 is too big, but closer than 200. 350?

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(350, 2, 15))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 0, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE              1932
2 TRUE               8068
```

Close! I reckon you could call that good (see below), or try again with a sample size a bit less than 350:

```r
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(345, 2, 15))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 0, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE              2124
2 TRUE               7876
```

340 is definitely too small:

```r
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(340, 2, 15))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 0, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE              2157
2 TRUE               7843
```

This is actually not as close as I was expecting. I think we are getting close to simulation accuracy for this number of simulations. If we do 10,000 simulations of an event with probability 0.8 (correctly rejecting this null), below are the kind of results we might get.[4] This is the middle 95% of that distribution.

```r
qbinom(c(0.025,0.975), 10000, 0.8)
```

[4] If the power is really 0.8, the number of simulated tests that end up rejecting has a binomial distribution with n of 10000 and p of 0.80.

```
[1] 7921 8078
```

Anything between those limits is the kind of thing we might get by chance, so simulation doesn't let us distinguish between 347 and 350 as the correct sample size. Unless we do more than 10,000 simulations, of course!

If you stuck with 1000 simulations each time, these are the corresponding limits:

```r
qbinom(c(0.025,0.975), 1000, 0.8)
```

```
[1] 775 824
```

and any sample sizes that produce an estimated power between these are as accurate as you'll get. (Here you see the advantage of doing more simulations.)

If you've been using 10,000 simulations each time like me, you'll have noticed that these actually take a noticeable time to run. This is why coders always have a coffee or something else to sip on while their code runs; coders, like us, need to see the output to decide what to do next. Or you could install the beepr package, and get some kind of sound when your simulation finishes, so that you'll know to get off Twitter[5] and see what happened. There are also packages that will send you a text message or will send a notification to all your devices.

What I want to see from you here is some kind of trial and error that proceeds logically, sensibly increasing or decreasing the sample size at each trial, until you have gotten reasonably close to power 0.8.

Extra: once again we can figure out the correct answer:

```r
power.t.test(power = 0.80, delta=2-0, sd=15, type="one.sample",
alternative = "one.sided")
```

```
     One-sample t test power calculation

              n = 349.1256
          delta = 2
             sd = 15
      sig.level = 0.05
          power = 0.8
    alternative = one.sided
```

[5]Or Reddit or Quora or whatever your favourite time-killer is.

This does not answer the question, though, since you need to do it by simulation with trial and error. If you want to do it this way, do it at the *end* as a check on your work; if the answer you get this way is very different from the simulation results, that's an invitation to check what you did.

350 actually *is* the correct answer. But you will need to try different sample sizes until you get close enough to a power of 0.8; simply doing it for $n = 350$ is not enough, because how did you know to try 350 and not some other sample size?

■

6.10 Power and α in a skewed population

A population of a large number of values `v` is at http://ritsokiguess.site/datafiles/pop.csv, in a CSV file.

(a) Read in the population and display some of the values.

Solution

```
my_url <- "http://ritsokiguess.site/datafiles/pop.csv"
pop <- read_csv(my_url)
```

```
Rows: 10000 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (1): v

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pop
```

```
# A tibble: 10,000 x 1
       v
   <dbl>
 1  9.97
 2  2.18
 3  6.20
 4  2.11
 5  6.30
```

```
 6  1.54
 7  5.77
 8  2.94
 9 16.8
10  1.95
# i 9,990 more rows
```

10,000 values. A large population. (From these few values, `v` seems to be positive but rather variable.)

■

(b) Obtain a suitable plot of your population. What do you notice?

Solution

One quantitative variable, so a histogram. The population is large, so you can use more bins than usual. Sturges' rule says 14 bins (the logarithm below is base 2, or, the next power of 2 above 10,000 is 16,384 which is 2^{14}):

```r
log(10000, 2)
```

```
[1] 13.28771
```

```r
2^14
```

```
[1] 16384
```

```r
ggplot(pop, aes(x=v)) + geom_histogram(bins=14)
```

Pick a number of bins: the default 30 bins is pretty much always too many. Any number of bins that shows this shape is good as an answer, but you also need to display some thinking about how many bins to use, either starting with a rule as I did, or experimenting with different numbers of bins. Rules are not hard and fast; it so happened that I liked the picture that 14 bins gave, so I stopped there. Thirty bins, the default, is actually not bad here:

```
ggplot(pop, aes(x=v)) + geom_histogram()
```

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

but if you do this, you need to say something that indicates some conscious thought, such as saying "this number of bins gives a good picture of the shape of the distribution", which I am OK with. *Have a reason for doing what you do.*

This is skewed to the right, or has a long right tail. This is a better description than "outliers": there are indeed some very large values (almost invisible on the histogram), but to say that is to imply that the rest of the distribution apart from the outliers has a regular shape, not something you can say here.[6]

Extra: The issue that's coming up is whether this is normally-distributed, which of course it is not. This is a normal quantile plot. (Idea: if the points follow the line, at least approximately, the variable is normally distributed; if not, not.):

```
ggplot(pop, aes(sample=v)) + stat_qq() + stat_qq_line()
```

[6]The question to ask yourself is whether the shape comes from the entire distribution, as it does here (skewness), or whether it comes from a few unusual observations (outliers).

That is your archetypal skewed-to-right. The two highest values are not a lot higher than the rest, again supporting a curved shape overall (skewness) on this plot, rather than the problem being outliers. (The highest values are consistent with the shape of the curve, rather than being unusually high compared to the curve.)

■

(c) If you take a sample of 10 observations from this population and run a t-test, how likely are you to (correctly) reject the null hypothesis $H_0 : \mu = 4$, against the alternative $H_a : \mu > 4$? Investigate by simulation.

Solution

As you noted, this is a one-sided alternative, so make sure your code does the right thing. Take a lot of random samples, run the t-test on each one, grab the P-value each time, count the number of P-values less or equal to your α. This is *not* a bootstrap, so the sampling needs to be *without* replacement, and you need to say how big the sample is:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(pop$v, 10))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 4, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE               830
2 TRUE                170
```

The estimated power is only about 0.19.

As to the code, well, the samples and the t-test both consist of more than one thing, so in the `mutate`s that create them, don't forget the `list` around the outside, which will create a list-column.

Here, and elsewhere in this question, use at least 1000 simulations. More will give you more accurate results, but you'll have to wait longer for it to run. Your choice.

As a final remark, you *can not* do this one by algebra, as you might have done in other courses, because you do not know the functional form of the population distribution. The power calculations you may have done before as calculations typically assume a normal population, because if you don't, the algebra gets too messy too fast. (You'd need to know the distribution of the test statistic under the *alternative* hypothesis, which in cases beyond the normal is not usually known.)

■

(d) Try again with a sample size of 50 (leaving everything else the same). Explain briefly why the results so far are as you'd expect.

Solution

For the code, this is copy-paste-edit. Just change the sample size:

```r
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(pop$v, 50))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 4, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE               262
2 TRUE                738
```

The power is now much bigger, around 73%. This is as expected because with a larger sample size we should be more likely to reject a false null hypothesis.

The reason for this is that the mean of a bigger sample should be closer to the population mean, because of the Law of Large Numbers, and thus further away from the incorrect null hypothesis and more likely far enough away to reject it. In this case, as you will see shortly, the population mean is 5, and so, with a bigger sample, the sample mean will almost certainly be closer to 5 and further away from 4.

I have a feeling you could formalize this kind of argument with Chebyshev's inequality, which would apply to any kind of population.[7] I think I'd have to write it down to get it right, though.

■

(e) Again by simulation, estimate the probability that the null hypothesis $H_0 : \mu = 5$ will be rejected when a sample of size 10 is taken from this population, in favour of the alternative $H_a : \mu > 5$. Explain briefly why the answer is not what you would have expected, and why that happened here. (Hint: what is the population mean?)

Solution

Taking the hint first:

```
pop %>%
summarize(m = mean(v))
```

```
# A tibble: 1 x 1
      m
  <dbl>
1  5.00
```

(I'm hoping that some light dawns at this point), and copy-paste-edit your simulation code again, this time changing the null mean to 5:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(pop$v, 10))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 5, alternative = "greater"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

[7] It has to have a standard deviation, though, but our population seems well-enough behaved to have a standard deviation.

```
# A tibble: 2 x 2
# Rowwise:
  `p_value <= 0.05`     n
  <lgl>             <int>
1 FALSE               980
2 TRUE                 20
```

The "power" is estimated to be 0.020. (Again, your value won't be exactly this, most likely, but it should be somewhere close.)

So what were we expecting? This time, the null hypothesis, that the population mean is 5, is actually *true*. So rejecting it is now a *type I error*, and the probability of that should be α, which was 0.05 here. In our simulation, though, the estimated probability is quite a bit *less* than 0.05. (Your result will probably differ from mine, but it is not likely to be bigger than 0.05).

To think about why that happened, remember that this is a very skewed population, and the sample size of 10 is not big, so this is not really the situation in which we should be using a t-test. The consequence of doing so anyway, which is what we investigated, is that the actual α of our test is not 0.05, but something smaller: the test is not properly calibrated.

If you do this again for a sample of size 50, you'll find that the simulation tells you that α is closer to 0.05, but still less. The population is skewed enough that the Central Limit Theorem still hasn't kicked in yet, and so we still cannot trust the t-test to give us a sensible P-value.

Extra: a lot more discussion on what is happening here:

This test is what is known in the jargon as "conservative". To a statistician, this means that the probability of making a type I error is smaller than it should be. That is in some sense safe, in that if you reject, you can be pretty sure that this rejection is correct, but it makes it a lot harder than it should to reject in the first place, and thus you can fail to declare a discovery when you have really made one (but the test didn't say so).

I did some investigation to see what was going on. First, I ran the simulation again, but this time keeping the mean and SD of each sample, as well as the t-statistic, but not actually doing the t-test:

```r
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(pop$v, 10))) %>%
  mutate(xbar = mean(my_sample),
         s = sd(my_sample),
         t_stat = (xbar - 5) / (s / sqrt(10))) -> mean_sd
mean_sd
```

```
# A tibble: 1,000 x 5
# Rowwise:
     sim my_sample   xbar     s  t_stat
   <int> <list>     <dbl> <dbl>   <dbl>
 1     1 <dbl [10]>  4.59  1.91 -0.685
 2     2 <dbl [10]>  3.82  2.03 -1.84
 3     3 <dbl [10]>  3.67  1.10 -3.84
 4     4 <dbl [10]>  6.62  1.77  2.90
 5     5 <dbl [10]>  6.77  5.07  1.10
 6     6 <dbl [10]>  9.01  4.78  2.65
 7     7 <dbl [10]>  3.38  2.80 -1.82
 8     8 <dbl [10]>  6.08  2.74  1.25
 9     9 <dbl [10]>  4.81  4.00 -0.151
10    10 <dbl [10]>  5.09  3.93  0.0736
# i 990 more rows
```

As for coding, I made a dataframe with a column `sim` that numbers the individual samples, made sure I said that I wanted to work rowwise, generated a random sample from the population in each row of size 10, and found its mean, SD and the calculated-by-me t-statistic.[8]

After that, I played around with several things, but I found something interesting when I plotted the sample mean and SD *against each other*:

```
ggplot(mean_sd, aes(x=xbar, y=s)) + geom_point() + geom_smooth(se=F)
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

[8]This is not bootstrapping, but generating ordinary random samples from a presumed-known population, so there is no `replace = TRUE` here.

When the sample mean is bigger, so is the sample standard deviation!

This actually does make sense, if you stop to think about it. A sample with a large mean will have some of those values from the long right tail in it, and having those values will also make the sample more spread out. The same does not happen at the low end: if the mean is small, all the sample values must be close together and the SD will be small also.[9]

It wasn't clear to me what that would do to the t-statistic. A larger sample mean would make the top of the test statistic bigger, but a larger sample mean would also go with a larger sample SD, and so the bottom of the test statistic would be bigger as well. That's why I included this in the simulation too:

```
ggplot(mean_sd, aes(x=t_stat)) + geom_histogram(bins=12)
```

[9]You might have something lurking in your mind about the sample mean and sample SD/variance being *independent*, which they clearly are not here. That is true if the samples come from a normal distribution, and from that comes independence of the top and bottom of the t-statistic. But here is an example of how everything fails once you go away from normality, and how you have to rely on the central limit theorem, or large sample sizes more generally, for most of your theory to be any good.

Well, well. Skewed to the *left*.

This too makes sense with a bit of thought. A small sample mean will also have a small sample SD, so the test statistic could be more negative. But a large sample mean will have a *large* sample SD, so the test statistic won't get so positive. Hence, in our simulation, the test statistic won't get large enough to reject with as often as it should. Thus, the type I error probability that is too small.

■

7 The sign test

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   0.5.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(smmr)
```

7.1 Running a maze

A researcher is trying to design a maze that can be run by rats in about 60 seconds. One particular maze was run by a sample of 21 rats, with the times shown in link.

(a) Read the data into R. What (if anything) are the data values delimited by?

(b) Run a sign test, doing it yourself as we did in class: count the number of values above and below 60, take the *smaller* of those, and find the probability of a value of that or smaller still on a binomial distribution with $n = 21$ and $p = 0.5$ (we have 21 data points), doubling the answer because the test is two-sided.

(c) Install my package `smmr`, if you haven't already. To do this, you first need to install the package `devtools` (if you haven't already), by going to the console and typing

```
install.packages("devtools")
```

When that's all done, install `smmr` thus:

```
library(devtools)
install_github("nxskok/smmr")
```

That all needs to be done only once. Then, each R Studio session where you want to use `smmr` needs this:

```
library(smmr)
```

As usual, only the `library` thing only needs to be done every time.

When you have `smmr` installed, use `sign_test` from that package to re-run your sign test. Do you get the same P-value?

(d) Package `smmr` also has a function `pval_sign`, which has the same input as `sign_test`, but with the null median *first*. Run it on your data and see what it gives.

(e) Obtain a 95% confidence interval for the median based on these data. Do this two ways. First, use the trial and error way from class (either the try-lots-of-values way or the bisection way; either is good). Second, use `ci_median` from `smmr`. The latter takes as input a data frame, a column name (unquoted) and optionally a `conf.level` that defaults to 0.95.

7.2 Chocolate chips

A famous cookie manufacturer claims that their bags of chocolate chip cookies contain "more than 1100 chocolate chips on average". A diligent group of students buys 16 bags of these cookies and counts the number of chocolate chips in each bag. The results are in http://ritsokiguess.site/datafiles/chips.txt.

(a) Read in and display (some of) the data.

(b) Build your own sign test in R for testing that the median is 1100 chocolate chips, against the alternative that it is greater. (Do this as in class: count the appropriate thing, compare it with an appropriate binomial distribution, and obtain a P-value.

(c) Use my R package `smmr` to reproduce your sign test above, and verify that you get consistent results. (See the maze-design question for instructions on installing this, if you haven't yet.)

(d) Use `smmr` to obtain a 95% confidence interval for the median number of chocolate chips per bag of cookies.

7.3 The power of the sign test

I've mentioned several times that the sign test has less power than the t-test. Let's investigate this with a specific example.

Let's suppose we are testing $H_0 : \mu = 40$ against $H_a : \mu \neq 40$, where μ is the population mean (and median, as we shall see). Our population actually has a normal distribution with mean 50 and SD 15, so that the null hypothesis is *wrong* and we want to reject it most of the time. On the other hand, the population actually *is* normally-distributed and so the t-test is the right one to use.

(This is an old question, so I tackle the simulated power differently than I did it in class this time. But see if you can follow what I do here.)

(a) Use `power.t.test` to find the probability that a t-test correctly rejects the null hypothesis using a sample size of $n = 10$.

(b) What code in R would draw a random sample of size 10 from the *true* population distribution and save the sample in a variable?

(c) What code would count how many of the sampled values are less than 40 and how many are greater (or equal)?

(d) It turns out the sign test would reject $H_0 : M = 40$ against $H_a : M \neq 40$ at $\alpha = 0.05$ if the smaller of the numbers in the last part is 1 or less. (M is the population median.) Add to your pipeline to obtain `TRUE` if you should reject the null for your data and `FALSE` otherwise.

(e) Simulate the above process 1000 times: draw a random sample from a normal distribution of size 10 with mean 50 and SD 15, count the number of values below 40, reject if the minimum of those is 0, 1, 9, or 10, then count the number of rejections out of 1000.

(f) Which is more powerful in this case, the sign test or the t-test? How do you know?

7.4 Ben Roethlisberger

Ben Roethlisberger plays (American) football for the Pittsburgh Steelers. He plays as a quarterback, which means that his job is to throw (pass) the ball so that one of his teammates can catch it. Each time he makes a pass that is caught, this is called a "completion", and the team coaches are interested in his average number of completions per game (this average could be the mean or the median).

In 2010, Roethlisberger was suspended for the first four games of the season, and there was concern that this might affect his performance (in terms of the number of passes completed in the games after he returned). The Pittsburgh Steelers did not play in week 5 of the 2010 season; the season is 17 weeks long (one game per week) and each team has one week in which they do not play.

The data are here. There are four columns: the year (always 2010), the week number of the season that the game was played in, the name of the opposing team, and the number of completed passes by Roethlisberger in the game.

(a) Read in and display (some of) the data. Do you have what you were expecting?

(b) Make a suitable graph of the number of completed passes, and explain briefly why you would have some doubts about using t-procedures in this situation.

(c) Run a sign test to compare Roethlisberger's performance in 2010 with his previous average of 22.5 completions per game. What do you conclude?

(d) Why might you have expected your sign test to come out significant, even without looking at the P-value? Explain briefly.

(e) Obtain a 90% confidence interval for the median number of completed passes (over "all possible games played by 2010 Ben Roethlisberger").

(f) Find a 90% confidence interval for the *mean* number of passes completed, and explain briefly why it differs from the one for the median in the way that it does.

7.5 Six ounces of protein

A company produces prepackaged diet meals. These meals are advertised as containing "6 ounces of protein per package". A consumer organization is concerned that this is not accurate. The organization takes a random sample of 20 of these meals, and measures the protein content of each one. The data are in http://ritsokiguess.site/datafiles/protein.txt as one column.

(a) Read in and display (some of) the data.

(b) Make a suitable graph of your data.

(c) Why might a sign test be better than a t-test for assessing the average amount of protein per package? Explain briefly. ("Average" here means any measure of centre.)

(d) Run a suitable sign test for these data. What do you conclude?

(e) In your sign test, how could you have deduced that the P-value was going to be small even without looking at any of the P-values themselves? Explain briefly.

(f) Obtain a 90% confidence interval for the population median protein content. What does this tell you about the reason for the rejection or non-rejection of the null hypothesis above?

My solutions follow:

7.6 Running a maze

A researcher is trying to design a maze that can be run by rats in about 60 seconds. One particular maze was run by a sample of 21 rats, with the times shown in link.

(a) Read the data into R. What (if anything) are the data values delimited by?

Solution

Take a look at the data file first. There is only one column of data, so you can treat it as being delimited by anything you like: a space, or a comma (the file can also be treated as a `.csv`), etc.:

```
myurl <- "http://ritsokiguess.site/datafiles/maze.txt"
times <- read_delim(myurl, " ")
```

```
Rows: 21 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (1): time

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
times
```

```
# A tibble: 21 x 1
    time
   <dbl>
 1  38.4
 2  46.2
 3  62.5
 4  38
 5  62.8
 6  33.9
```

```
 7  50.4
 8  35
 9  52.8
10  60.1
# i 11 more rows
```

■

(b) Run a sign test, doing it yourself as we did in class: count the number of values above and below 60, take the *smaller* of those, and find the probability of a value of that or smaller still on a binomial distribution with $n = 21$ and $p = 0.5$ (we have 21 data points), doubling the answer because the test is two-sided.

Solution

Count how many values are above and below 60:

```
times %>% count(time > 60)
```

```
# A tibble: 2 x 2
  `time > 60`     n
  <lgl>       <int>
1 FALSE          16
2 TRUE            5
```

5 above and 16 below. Then find out how likely it is that a binomial with $n = 21, p = 0.5$ would produce 5 or fewer successes:

```
p <- sum(dbinom(0:5, 21, 0.5))
p
```

```
[1] 0.01330185
```

or if you prefer count upwards from 16:

```
sum(dbinom(16:21, 21, 0.5))
```

```
[1] 0.01330185
```

and double it to get a two-sided P-value:

```
2 * p
```

```
[1] 0.0266037
```

We'll compare this with `smmr` in a moment.

■

(c) Install my package `smmr`, if you haven't already. To do this, you first need to install the package `devtools` (if you haven't already), by going to the console and typing

```
install.packages("devtools")
```

When that's all done, install `smmr` thus:

```
library(devtools)
install_github("nxskok/smmr")
```

That all needs to be done only once. Then, each R Studio session where you want to use `smmr` needs this:

```
library(smmr)
```

As usual, only the `library` thing only needs to be done every time.

When you have `smmr` installed, use `sign_test` from that package to re-run your sign test. Do you get the same P-value?

Solution

The sign test function takes a data frame, an (unquoted) column name from that data frame of data to test the median of, and a null median (which defaults to 0 if you omit it):

```
library(smmr)
sign_test(times, time, 60)
```

```
$above_below
below above
   16     5

$p_values
  alternative    p_value
```

```
1       lower 0.01330185
2       upper 0.99640131
3   two-sided 0.02660370
```

This shows you two things: a count of the values below and above the null median, and then the P-values according to the various alternative hypotheses you might have.

In our case, we see again the 16 maze-running times below 60 seconds and 5 above (one of which was a long way above, but we don't care about that here). We were testing whether the median was different from 60, so we look at the two-sided P-value of 0.0266, which is exactly what we had before.

If `sign_test` doesn't work for you (perhaps because it needs a function `enquo` that you don't have), there is an alternative function `sign_test0` that doesn't use it. It requires as input a *column* of values (extracted from the data frame) and a null median, thus:

```r
with(times, sign_test0(time, 60))
```

```
$above_below
below above
   16     5

$p_values
  alternative    p_value
1       lower 0.01330185
2       upper 0.99640131
3   two-sided 0.02660370
```

The output should be, and here is, identical.

■

(d) Package `smmr` also has a function `pval_sign`, which has the same input as `sign_test`, but with the null median *first*. Run it on your data and see what it gives.

Solution

Try it and see:

```r
pval_sign(60, times, time)
```

```
[1] 0.0266037
```

The two-sided P-value, and that is all. We'll be using this in a minute.

Alternatively, there is also this, which needs a null median and a *column* as input:

```
with(times, pval_sign0(60, time))
```

```
[1] 0.0266037
```

■

(e) Obtain a 95% confidence interval for the median based on these data. Do this two ways. First, use the trial and error way from class (either the try-lots-of-values way or the bisection way; either is good). Second, use `ci_median` from `smmr`. The latter takes as input a data frame, a column name (unquoted) and optionally a `conf.level` that defaults to 0.95.

Solution

The reason for showing you `pval_sign` in the previous part is that this is a building block for the confidence interval. What we do is to try various null medians and find out which ones give P-values less than 0.05 (outside the interval) and which ones bigger (inside). We know that the value 60 is outside the 95% CI, and the sample median is close to 50 (which we expect to be inside), so sensible values to try for the upper end of the interval would be between 50 and 60:

```
pval_sign(58, times, time)
```

```
[1] 0.0266037
```

```
pval_sign(55, times, time)
```

```
[1] 0.6636238
```

So, 55 is inside the interval and 58 is outside. I could investigate further in similar fashion, but I thought I would try a whole bunch of null medians all at once. That goes like this, `rowwise` because `pval_sign` expects *one* null-hypothesis median, not several all at once:

```
tibble(meds = seq(55, 58, 0.25)) %>%
  rowwise() %>%
  mutate(pvals = pval_sign(meds, times, time))
```

```
# A tibble: 13 x 2
# Rowwise:
    meds  pvals
   <dbl>  <dbl>
 1  55   0.664
 2  55.2 0.383
 3  55.5 0.263
 4  55.8 0.189
 5  56   0.189
 6  56.2 0.189
 7  56.5 0.0784
 8  56.8 0.0784
 9  57   0.0784
10  57.2 0.0784
11  57.5 0.0784
12  57.8 0.0266
13  58   0.0266
```

So values for the median all the way up to and including 57.5 are in the confidence interval.

Now for the other end of the interval. I'm going to do this a different way: more efficient, but less transparent. The first thing I need is a pair of values for the median: one inside the interval and one outside. Let's try 40 and 50:

```r
pval_sign(40, times, time)
```

```
[1] 0.00719738
```

```r
pval_sign(50, times, time)
```

```
[1] 1
```

OK, so 40 is outside and 50 is inside. So what do I guess for the next value to try? I could do something clever like assuming that the relationship between hypothesized median and P-value is *linear*, and then guessing where that line crosses 0.05. But I'm going to assume *nothing* about the relationship except that it goes uphill, and therefore crosses 0.05 somewhere. So my next guess is halfway between the two values I tried before:

```r
pval_sign(45, times, time)
```

```
[1] 0.07835388
```

So, 45 is inside the interval, and my (slightly) improved guess at the bottom end of the interval is that it's between 40 and 45. So next, I try halfway between *those*:

```r
pval_sign(42.5, times, time)
```

```
[1] 0.0266037
```

42.5 is outside, so the bottom end of the interval is between 42.5 and 45.

What we are doing is narrowing down where the interval's bottom end is. We started by knowing it to within 10, and now we know it to within 2.5. So if we keep going, we'll know it as accurately as we wish.

This is called a "bisection" method, because at each step, we're dividing our interval by 2.

There is one piece of decision-making at each step: if the P-value for the median you try is greater than 0.05, that becomes the top end of your interval (as when we tried 45); if it is less, it becomes the bottom end (when we tried 42.5).

This all begs to be automated into a loop. It's not a `for`-type loop, because we don't know how many times we'll be going around. It's a `while` loop: keep going while something is true. Here's how it goes:

```r
lo <- 40
hi <- 50
while (abs(hi - lo) > 0.1) {
  try <- (hi + lo) / 2
  ptry <- pval_sign(try, times, time)
  print(c(try, ptry))
  if (ptry < 0.05) {
    lo <- try
  } else {
    hi <- try
  }
}
```

```
[1] 45.00000000  0.07835388
[1] 42.5000000  0.0266037
[1] 43.7500000  0.0266037
[1] 44.37500000  0.07835388
```

```
[1] 44.0625000  0.0266037
[1] 44.2187500  0.0266037
[1] 44.2968750  0.0266037
```

```
lo
```

```
[1] 44.29688
```

```
pval_sign(lo, times, time)
```

```
[1] 0.0266037
```

```
hi
```

```
[1] 44.375
```

```
pval_sign(hi, times, time)
```

```
[1] 0.07835388
```

The loop stopped because 44.297 and 44.375 are less than 0.1 apart. The first of those is outside the interval and the second is inside. So the bottom end of our interval is 44.375, to this accuracy. If you want it more accurately, change 0.1 in the `while` line to something smaller (but then you'll be waiting longer for the answer).

I put the `print` statement in the loop so that you could see what values were being tried, and what P-values they were producing. What happens with these is that the P-value jumps at each data value, so you won't get a P-value exactly 0.05; you'll get one above and one below.

Likewise, you can use the function with a zero on its name and feed it a column rather than a data frame and a column name:

```
tibble(meds =  seq(55, 58, 0.25)) %>%
  rowwise() %>%
  mutate(pvals =  with(times, pval_sign0(meds, time)))
```

```
# A tibble: 13 x 2
# Rowwise:
    meds  pvals
   <dbl>  <dbl>
 1  55   0.664
 2  55.2 0.383
 3  55.5 0.263
 4  55.8 0.189
 5  56   0.189
 6  56.2 0.189
 7  56.5 0.0784
 8  56.8 0.0784
 9  57   0.0784
10  57.2 0.0784
11  57.5 0.0784
12  57.8 0.0266
13  58   0.0266
```

Or adapt the idea I had above for bisection. All that was a lot of work, but I wanted you to see it all once, so that you know where the confidence interval is coming from. `smmr` also has a function `ci_median` that does all of the above without you having to do it. As I first wrote it, it was using the trial and error thing with `rowwise`, but I chose to rewrite it with the bisection idea, because I thought that would be more accurate.

```
ci_median(times, time)
```

```
[1] 44.30747 57.59766
```

This is a more accurate interval than we got above. (The `while` loop for the bisection keeps going until the two guesses at the appropriate end of the interval are less than 0.01 apart, by default.)[1]

If you want some other confidence level, you add `conf.level` on the end, as you would for `t.test`:

```
ci_median(times, time, conf.level = 0.75)
```

```
[1] 46.20444 55.49473
```

[1]You can change this by adding something like *tol=1e-4* to the end of your *ci-median*.

A 75% CI, just for fun. This is a shorter interval than the 95% one, as it should be.

Likewise there is a `ci_median0` that takes a column and an optional confidence level:

```
with(times, ci_median0(time))
```

```
[1] 44.30747 57.59766
```

```
with(times, ci_median0(time, conf.level = 0.75))
```

```
[1] 46.20444 55.49473
```

with the same results. Try `ci_median` first, and if it doesn't work, try `ci_median0`.

■

7.7 Chocolate chips

A famous cookie manufacturer claims that their bags of chocolate chip cookies contain "more than 1100 chocolate chips on average". A diligent group of students buys 16 bags of these cookies and counts the number of chocolate chips in each bag. The results are in http://ritsokiguess.site/datafiles/chips.txt.

(a) Read in and display (some of) the data.

Solution

I'll pretend it's a `.csv` this time, just for fun. Give the data frame a name different from `chips`, so that you don't get confused:

```
my_url <- "http://ritsokiguess.site/datafiles/chips.txt"
bags <- read_csv(my_url)
```

```
Rows: 16 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (1): chips

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
bags
```

```
# A tibble: 16 x 1
   chips
   <dbl>
 1  1219
 2  1214
 3  1087
 4  1200
 5  1419
 6  1121
 7  1325
 8  1345
 9  1244
10  1258
11  1356
12  1132
13  1191
14  1270
15  1295
16  1135
```

That looks sensible.

■

(b) Build your own sign test in R for testing that the median is 1100 chocolate chips, against the alternative that it is greater. (Do this as in class: count the appropriate thing, compare it with an appropriate binomial distribution, and obtain a P-value.

Solution

The null median is 1100, so we count the number of values above and below:

```r
bags %>% count(chips<1100)
```

```
# A tibble: 2 x 2
  `chips < 1100`     n
  <lgl>          <int>
1 FALSE             15
2 TRUE               1
```

The un-standard thing there is that we can put a logical condition directly into the `count`. If you don't think of that, you can also do this, which creates a new variable `less` that is `TRUE` or `FALSE` for each bag appropriately:

```
bags %>% mutate(less=(chips<1100)) %>% count(less)
```

```
# A tibble: 2 x 2
  less      n
  <lgl> <int>
1 FALSE    15
2 TRUE      1
```

or the more verbose

```
bags %>% mutate(less=(chips<1100)) %>%
group_by(less) %>% summarize(howmany=n())
```

```
# A tibble: 2 x 2
  less  howmany
  <lgl>   <int>
1 FALSE      15
2 TRUE        1
```

Just one value below, with all the rest above. Getting the right P-value, properly, requires some careful thought (but you will probably get the right answer anyway). If the alternative hypothesis is true, and the median is actually bigger than 1100 (say, 1200), you would expect half the data values to be bigger than 1200 and half smaller. So *more* than half the data values would be bigger than *1100*, and fewer than half of them would be less than 1100. So, if we are going to reject the null (as it looks as if we will), that small number of values below 1100 is what we want.

The P-value is the probability of a value 1 or less in a binomial distribution with $n = 16, p = 0.5$:

```
sum(dbinom(0:1,16,0.5))
```

```
[1] 0.0002593994
```

Or, equivalently, count *up* from 15:

```r
sum(dbinom(15:16,16,0.5))
```

```
[1] 0.0002593994
```

This is correctly one-sided, so we don't have to do anything with it.

■

(c) Use my R package **smmr** to reproduce your sign test above, and verify that you get consistent results. (See the maze-design question for instructions on installing this, if you haven't yet.)

Solution

This will mean reading the output carefully:

```r
library(smmr)
sign_test(bags,chips,1100)
```

```
$above_below
below above
    1    15

$p_values
  alternative      p_value
1       lower 0.9999847412
2       upper 0.0002593994
3   two-sided 0.0005187988
```

This time, we're doing a one-sided test, specifically an *upper-tail* test, since we are looking for evidence that the median is *greater than* 1100. The results are exactly what we got "by hand": 15 values above and one below, and a P-value (look along the `upper` line) of 0.00026. The two-sided P-value of 0.00052 rounds to the same 0.0005 as SAS got.

Alternatively, you can do this:

```r
sign_test0(bags$chips,1100)
```

```
$above_below
below above
    1    15
```

```
$p_values
  alternative      p_value
1       lower 0.9999847412
2       upper 0.0002593994
3   two-sided 0.0005187988
```

with the same result (but only go this way if you need to).

■

(d) Use `smmr` to obtain a 95% confidence interval for the median number of chocolate chips per bag of cookies.

Solution

Once everything is in place, this is simplicity itself:

```
ci_median(bags,chips)
```

```
[1] 1135.003 1324.996
```

1135 to 1325. I would round these off to whole numbers, since the data values are all whole numbers. These values are all above 1100, which supports the conclusion we got above that the median is above 1100. This is as it should be, because the CI is "all those medians that would *not* be rejected by the sign test".

Or,

```
ci_median0(bags$chips)
```

```
[1] 1135.003 1324.996
```

■

7.8 The power of the sign test

I've mentioned several times that the sign test has less power than the t-test. Let's investigate this with a specific example.

Let's suppose we are testing $H_0 : \mu = 40$ against $H_a : \mu \neq 40$, where μ is the population mean (and median, as we shall see). Our population actually has a normal distribution with mean 50 and SD 15, so that the null hypothesis is *wrong* and we want to reject it most of the time. On the other hand, the population actually *is* normally-distributed and so the t-test is the right one to use.

(This is an old question, so I tackle the simulated power differently than I did it in class this time. But see if you can follow what I do here.)

(a) Use `power.t.test` to find the probability that a t-test correctly rejects the null hypothesis using a sample size of $n = 10$.

Solution

```
power.t.test(delta=50-40,n=10,sd=15,type="one.sample",alternative="two.sided")
```

```
     One-sample t test power calculation

              n = 10
          delta = 10
             sd = 15
      sig.level = 0.05
          power = 0.4691805
    alternative = two.sided
```

The power is 0.469. Not great, but we'll see how this stacks up against the sign test.

■

(b) What code in R would draw a random sample of size 10 from the *true* population distribution and save the sample in a variable?

Solution

The data actually have a normal distribution with mean 50 and SD 15, so we use `rnorm` with this mean and SD, obtaining 10 values:

```r
x=rnorm(10,50,15)
x
```

```
 [1] 57.39643 71.76733 29.30209 33.05134 63.91260 59.81963 27.34607 50.85185
 [9] 46.01039 57.46783
```

■

(c) What code would count how many of the sampled values are less than 40 and how many are greater (or equal)?

Solution

The way we know this is to put `x` into a data frame first:

```r
tibble(x) %>% count(x<40)
```

```
# A tibble: 2 x 2
  `x < 40`     n
  <lgl>    <int>
1 FALSE        7
2 TRUE         3
```

2 values less (and 8 greater-or-equal).

■

(d) It turns out the sign test would reject $H_0 : M = 40$ against $H_a : M \neq 40$ at $\alpha = 0.05$ if the smaller of the numbers in the last part is 1 or less. (M is the population median.) Add to your pipeline to obtain `TRUE` if you should reject the null for your data and `FALSE` otherwise.

Solution

This is actually easier than you might think. The output from `count` is a data frame with a column called `n`, whose minimum value you want. I add to my pipeline:

```r
tibble(x) %>% count(x<40) %>%
summarize(the_min=min(n)) %>%
mutate(is_rejected=(the_min<=1))
```

```
# A tibble: 1 x 2
  the_min is_rejected
    <int> <lgl>
1       3 FALSE
```

This will fail sometimes. If all 10 of your sample values are greater than 40, which they might turn out to be, you'll get a table with only one line, `FALSE` and 10; the minimum of the `n` values is 10 (since there is only one), and it will falsely say that you should not reject. The fix is

```
tibble(x) %>% count(x<40) %>%
summarize(the_min=min(n)) %>%
mutate(is_rejected=(the_min<=1 | the_min==10))
```

```
# A tibble: 1 x 2
  the_min is_rejected
    <int> <lgl>
1       3 FALSE
```

The above is almost the right thing, but not quite: we only want that value that I called `is_rejected`, rather than the whole data frame, so a `pull` will grab it:

```
tibble(x) %>% count(x<40) %>%
summarize(the_min=min(n)) %>%
mutate(is_rejected=(the_min<=1 | the_min==10)) %>%
pull(is_rejected)
```

```
[1] FALSE
```

You might be wondering where the "1 or less" came from. Getting a P-value for the sign test involves the binomial distribution: if the null is correct, each data value is independently either above or below 40, with probability 0.5 of each, so the number of values below 40 (say) is binomial with $n = 10$ and $p = 0.5$. The P-value for 1 observed value below 40 and the rest above is

```
2*pbinom(1,10,0.5)
```

```
[1] 0.02148438
```

which is less than 0.05; the P-value for 2 values below 40 and the rest above is

```r
2*pbinom(2,10,0.5)
```

```
[1] 0.109375
```

which is bigger than 0.05.

You might have encountered the term "critical region" for a test. This is the values of your test statistic that you would reject the null hypothesis for. In this case, the critical region is 1 and 0 observations below 40, along with 1 and 0 observations above 40.

When you're thinking about power, I think it's easiest to think in terms of the critical region (rather than directly in terms of P-values) since you have a certain α in mind all the way through, 0.05 in the power examples that I've done. The steps are then:

- Work out the critical region for your test, the values of the test statistic (or sample mean or sample count) that would lead to rejecting the null hypothesis.
- Under your particular alternative hypothesis, find the probability of falling into your critical region.

When I say "work out", I mean either calculating (along the lines of STAB57), or simulating, as we have done here.

■

(e) Simulate the above process 1000 times: draw a random sample from a normal distribution of size 10 with mean 50 and SD 15, count the number of values below 40, reject if the minimum of those is 0, 1, 9, or 10, then count the number of rejections out of 1000.

Solution

Set up a dataframe with a column (called, maybe, `sim`) that counts the number of simulations you are doing, and then use `rowwise` to take a random sample in each row and extract what you need from it.

I start with setting the random number seed, so it comes out the same each time. That way, if I rerun the code, my answers are the same (and I don't have to change my discussion of them.)

```r
set.seed(457299)
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rnorm(10, 50, 15)))
```

```
# A tibble: 1,000 x 2
# Rowwise:
     sim sample
   <int> <list>
 1     1 <dbl [10]>
 2     2 <dbl [10]>
 3     3 <dbl [10]>
 4     4 <dbl [10]>
 5     5 <dbl [10]>
 6     6 <dbl [10]>
 7     7 <dbl [10]>
 8     8 <dbl [10]>
 9     9 <dbl [10]>
10    10 <dbl [10]>
# i 990 more rows
```

Each sample has 10 values in it, not just one, so you need the `list` around the `rnorm`. Note that `sample` is labelled as a list-column.

Now we have to count how many of the sample values are less than 40:

```r
set.seed(457299)
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rnorm(10, 50, 15))) %>%
  mutate(less = list(sample<40)) %>%
  mutate(counted = sum(less))
```

```
# A tibble: 1,000 x 4
# Rowwise:
     sim sample     less       counted
   <int> <list>     <list>       <int>
 1     1 <dbl [10]> <lgl [10]>       3
 2     2 <dbl [10]> <lgl [10]>       4
 3     3 <dbl [10]> <lgl [10]>       2
 4     4 <dbl [10]> <lgl [10]>       1
 5     5 <dbl [10]> <lgl [10]>       4
 6     6 <dbl [10]> <lgl [10]>       2
 7     7 <dbl [10]> <lgl [10]>       1
 8     8 <dbl [10]> <lgl [10]>       1
 9     9 <dbl [10]> <lgl [10]>       2
10    10 <dbl [10]> <lgl [10]>       5
# i 990 more rows
```

This is a bit of a programmer's trick. In R, `less` contains a vector of 10 TRUE or FALSE values, according to whether the corresponding value in `sample` is less than 40 or not. In R (and many other programming languages), the numeric value of TRUE is 1 and of FALSE is 0, so you count how many TRUE values there are by adding them up. To verify that this worked, we should `unnest` `sample` and `less`:

```
set.seed(457299)
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rnorm(10, 50, 15))) %>%
  mutate(less = list(sample<40)) %>%
  mutate(counted = sum(less)) %>%
  unnest(c(sample, less))
```

```
# A tibble: 10,000 x 4
     sim sample less  counted
   <int>  <dbl> <lgl>   <int>
 1     1   74.3 FALSE       3
 2     1   38.8 TRUE        3
 3     1   46.0 FALSE       3
 4     1   39.5 TRUE        3
 5     1   53.2 FALSE       3
 6     1   60.6 FALSE       3
 7     1   33.8 TRUE        3
 8     1   61.9 FALSE       3
 9     1   50.1 FALSE       3
10     1   66.4 FALSE       3
# i 9,990 more rows
```

In the first sample, 38.8, 39.5, and 33.8 are less than 40, correctly identified so in `less`, and the `counted` column shows that the first sample did indeed have 3 values less than 40. You can check a few of the others as well, enough to convince yourself that this is working.

Next, the sign test will reject if there are 0, 1, 9 or 10 values less than 40 (you might be guessing that the last two will be pretty unlikely), so make a column called `reject` that encapsulates that, and then count how many times you rejected in your simulations. I don't need my `unnest` any more; that was just to check that everything was working so far:

```
set.seed(457299)
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rnorm(10, 50, 15))) %>%
```

```
  mutate(less = list(sample<40)) %>%
  mutate(counted = sum(less)) %>%
  mutate(reject = (counted<=1 | counted >= 9)) %>%
  count(reject)
```

```
# A tibble: 2 x 2
# Rowwise:
  reject     n
  <lgl>  <int>
1 FALSE    757
2 TRUE     243
```

My simulated power is 0.243

This is all liable to go wrong the first few times, so make sure that each line works before you go on to the next, as I did. While you're debugging, try it with a small number of random samples like 5. (It is smart to have a variable called `nsim` which you set to a small number like 5 when you are testing, and than to 1000 when you run the real thing, so that the first line of the pipeline is then `tibble(sim = 1:nsim)`.)

If you were handing something like this in, I would only want to see your code for the final pipeline that does everything, though you could and should have some words that describe what you did.

I'm now thinking a better way to do this is to write a function that takes a sample (in a vector) and returns a TRUE or FALSE according to whether or not a median of 40 would be rejected for that sample:

```
is_reject=function(x) {
  tibble(x=x) %>%
    mutate(counted = (x < 40)) %>%
    summarize(below = sum(counted)) %>%
    summarize(is_rejected = (below<=1 | below>=9)) %>%
    pull(is_rejected)
}
is_reject(c(35, 45, 55))
```

```
[1] TRUE
```

```
is_reject(c(35, 38, 45, 55))
```

```
[1] FALSE
```

Now, we have to use that:

```
set.seed(457299)
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rnorm(10, 50, 15))) %>%
  mutate(reject = is_reject(sample)) %>%
  count(reject)
```

```
# A tibble: 2 x 2
# Rowwise:
  reject     n
  <lgl>  <int>
1 FALSE    757
2 TRUE     243
```

This is a bit cleaner because the process of deciding whether each sample leads to a rejection of the median being 40 has been "outsourced" to the function, and the pipeline with the `rowwise` is a lot cleaner: take a sample, decide whether that sample leads to rejection, and count up the rejections.

■

(f) Which is more powerful in this case, the sign test or the t-test? How do you know?

Solution

The power of the sign test is estimated as 0.243, which is quite a bit less than the power of the t-test, which we found back in (a) to be 0.469. So the t-test, in this situation where it is valid, is the right test to use: it is (i) valid and (ii) more powerful. So the t-test is more powerful. One way to think about how *much* more powerful is to ask "how much smaller of a sample size would be needed for the t-test to have the same power as this sign test?" The power of my sign test was 0.243, so in `power.t.test` we set `power` equal to that and omit the sample size `n`:

```
power.t.test(delta=50-40,power=0.243,sd=15,type="one.sample",alternative="two.sided")
```

```
     One-sample t test power calculation
```

```
             n = 5.599293
         delta = 10
            sd = 15
     sig.level = 0.05
         power = 0.243
   alternative = two.sided
```

A sample of size 6 gives the same power for the t-test that a sample of size 10 does for the sign test. The ratio of these two sample sizes is called the *relative efficiency* of the two tests: in this case, the t-test is $10/6 = 1.67$ times more efficient. The data that you have are being used "more efficiently" by the t-test. It is possible to derive[2]
the limiting relative efficiency of the t test relative to the sign test when the data are actually normal, as the sample size gets larger. This turns out not to depend on how far wrong the null is (as long as it is the same for both the t-test and the sign test). This "asymptotic relative efficiency" is $\pi/2 = 1.57$. Our relative efficiency for power 0.243, namely 1.67, was pretty close to this, even though our sample sizes 10 and 6 are not especially close to infinity. This says that, if your data are actually from a normal distribution, you do a lot better to use the t-test than the sign test, because the sign test is wasteful of data (it only uses above/below rather than the actual values).

Extra: if your data are *not* from a normal distribution, then the story can be very different. Of course you knew I would investigate this. There is a distribution called the "Laplace" or "double exponential" distribution, that has very long tails.[3] The distribution is not in base R, but there is a package called `smoothmest` that contains a function `rdoublex` to generate random values from this distribution. So we're going to do a simulation investigation of the power of the sign test for Laplace data, by the same simulation technique that we did above. Like the normal, the Laplace distribution is symmetric, so its mean and median are the same (which makes our life easier).[4]

Let's test the hypothesis that the median is zero. We'll suppose that the true median is 0.5 (this is called `mu` in `rdoublex`). The first problem we run into is that we can't use `power.t.test` because they assume normal data, which we are far from having. So we have to do two simulations: one to simulate the power of the t test, and one to simulate the power of the sign test.

To simulate the t test, we first have to generate some Laplace data with the true mean of 0.5. We'll use a sample size of 50 throughout these simulations.

[2]Meaning, I forget how to do it. But it has something to do with looking at alternatives that are very close to the null.

[3]If you've ever run into the exponential distribution, you'll recall that this is right skewed with a very long tail. The Laplace distribution looks like two of these glued back to back.

[4]This is about the *only* way in which the normal and Laplace distributions are alike.

```r
library(smoothmest)
rl <- rdoublex(50,mu=0.5)
rl
```

```
 [1] -0.33323285  0.70569291 -1.22513053  0.68517708  0.87221482  0.49250051
 [7]  0.26700527  1.90236874  0.53288312  1.37374732  0.72743434  0.46634071
[13]  0.43581431 -0.01545866  0.18594908 -0.40403202 -0.13540289  0.83862694
[19] -0.23360644 -0.74050354  2.92089551 -2.72173880  0.51571185  1.23636045
[25]  0.82921382  1.72456334  0.07903058  0.74789589  0.90487190  2.52310082
[31]  3.13629814  0.81851434  0.74615575 -0.26068744  2.70683355  1.46981530
[37]  1.45646489  1.20232517  6.65249860 -0.51575026 -0.07606399  2.11338640
[43] -1.20427995  1.70986104 -1.66466321  0.55346854  0.33908531  0.72100677
[49]  0.92025176  0.98922656
```

This seems to have some unusual values, far away from zero:

```r
tibble(rl) %>%
ggplot(aes(sample=rl))+
stat_qq()+stat_qq_line()
```

You see the long tails compared to the normal.

Now, we feed these values into `t.test` and see whether we reject a null median of zero (at $\alpha = 0.05$):

```
tt <- t.test(rl)
tt
```

```

    One Sample t-test

data:  rl
t = 3.72, df = 49, p-value = 0.0005131
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.3399906 1.1388911
sample estimates:
mean of x
0.7394408
```

Or we can just pull out the P-value and even compare it to 0.05:

```
pval <- tt$p.value
pval
```

```
[1] 0.0005130841
```

```
is.reject <- (pval<=0.05)
is.reject
```

```
[1] TRUE
```

This one has a small P-value and so the null median of 0 should be (correctly) rejected.

We'll use these ideas to simulate the power of the t-test for these data, testing a mean of 0. This uses the same ideas as for any power simulation; the difference here is the true distribution:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rdoublex(50, mu = 0.5))) %>%
  mutate(t_test = list(t.test(sample, mu = 0))) %>%
```

```r
  mutate(t_pval = t_test$p.value) %>%
  count(t_pval <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `t_pval <= 0.05`     n
  <lgl>            <int>
1 FALSE              298
2 TRUE               702
```

And now we simulate the sign test. Since what we want is a P-value from a vector, the easiest way to do this is to use `pval_sign0` from `smmr`, which returns exactly the two-sided P-value that we want, so that the procedure is a step simpler:

```r
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(rdoublex(50, mu = 0.5))) %>%
  mutate(sign_pval = pval_sign0(0, sample)) %>%
  count(sign_pval <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `sign_pval <= 0.05`     n
  <lgl>               <int>
1 FALSE                 228
2 TRUE                  772
```

For data from this Laplace distribution, the power of this t-test is 0.696, but the power of the sign test on the same data is 0.761, *bigger*. For Laplace-distributed data, the sign test is *more* powerful than the t-test.

This is not to say that you will ever run into data that comes from the Laplace distribution. But the moral of the story is that the sign test *can* be more powerful than the t-test, under the right circumstances (and the above simulation is the "proof" of that statement). So a blanket statement like "the sign test is not very powerful" needs to be qualified a bit: when your data come from a sufficiently long-tailed distribution, the sign test can be more powerful relative to the t-test than you would think.

■

7.9 Ben Roethlisberger

Ben Roethlisberger plays (American) football for the Pittsburgh Steelers. He plays as a quarterback, which means that his job is to throw (pass) the ball so that one of his teammates can catch it. Each time he makes a pass that is caught, this is called a "completion", and the team coaches are interested in his average number of completions per game (this average could be the mean or the median).

In 2010, Roethlisberger was suspended for the first four games of the season, and there was concern that this might affect his performance (in terms of the number of passes completed in the games after he returned). The Pittsburgh Steelers did not play in week 5 of the 2010 season; the season is 17 weeks long (one game per week) and each team has one week in which they do not play.

The data are here. There are four columns: the year (always 2010), the week number of the season that the game was played in, the name of the opposing team, and the number of completed passes by Roethlisberger in the game.

(a) Read in and display (some of) the data. Do you have what you were expecting?

Solution

Reading in is the usual, noting that this is a `.csv`:

```
my_url <- "http://ritsokiguess.site/datafiles/roethlisberger.csv"
ben <- read_csv(my_url)
```

```
Rows: 12 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): opponent
dbl (3): season, week, completed

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ben
```

```
# A tibble: 12 x 4
   season  week opponent      completed
    <dbl> <dbl> <chr>             <dbl>
 1   2010     6 Cleveland            16
```

```
 2   2010     7 Miami               19
 3   2010     8 New Orleans         17
 4   2010     9 Cincinnati          17
 5   2010    10 New England         30
 6   2010    11 Oakland             18
 7   2010    12 Buffalo             20
 8   2010    13 Baltimore           22
 9   2010    14 Cincinnati          21
10   2010    15 New York Jets       23
11   2010    16 Carolina            22
12   2010    17 Cleveland           15
```

Since "Roethlisberger" is a lot to type every time, I called the dataframe by his first name.

I am showing all 12 rows here; you are probably seeing only 10, and will have to scroll down to see the last two.

I have the four variables promised, and I also have a sensible number of rows. In particular, there is no data for weeks 1–4 (the suspension) and for week 5 (in which the team did not play), but there is a number of passes completed for all the other weeks of the season up to week 17. (If Roethlisberger had not played in any other games, you can expect that I would have told you about it.)

Extra: I did some processing to get the data to this point. I wanted to ask you about the 2010 season, and that meant having the 2009 data to compare it with. So I went here, scrolled down to Schedule and Game Results, and clicked on each of the Boxscores to get the player stats by game. Then I made a note of the opponent and the number of passes completed, and did the same for 2010. I put them in a file I called `r1.txt`, in aligned columns, and read that in. (An alternative would have been to make a spreadsheet and save that as a `.csv`, but I already had R Studio open.) Thus:

```
r0 <- read_table("r1.txt")
```

```
-- Column specification --------------------------------------------------------
cols(
  season = col_double(),
  week = col_double(),
  opponent = col_character(),
  completed = col_double()
)
```

```
r0
```

```
# A tibble: 27 x 4
   season  week opponent    completed
    <dbl> <dbl> <chr>           <dbl>
 1   2009     1 Tennessee          33
 2   2009     2 Chicago            23
 3   2009     3 Cincinnati         22
 4   2009     4 San_Diego          26
 5   2009     5 Detroit            23
 6   2009     6 Cleveland          23
 7   2009     7 Minnesota          14
 8   2009     9 Denver             21
 9   2009    10 Cincinnati         20
10   2009    11 Kansas_City        32
# i 17 more rows
```

I was curious about the season medians (for reasons you see later), thus:

```
r0 %>% group_by(season) %>% summarise(med = median(completed))
```

```
# A tibble: 2 x 2
  season   med
   <dbl> <dbl>
1   2009  22
2   2010  19.5
```

You will realize that my asserted average for "previous seasons" is close to the median for 2009. Here is where I have to admit that I cheated. It actually *is* the median for 2009, except that there are some games in 2010 where Roethlisberger had 22 completed passes and I didn't want to mess the sign test up (I talk more about this later). So I made it 22.5, which is a possible value for the median of an even number of whole-number values.

Anyway, the last thing to do is to grab only the rows for 2010 and save them for you. This uses `filter` to select only the rows for which something is true:

```
library(smmr)
r0 %>% filter(season==2010) -> r1
write_csv(r1, "roethlisberger.csv")
```

■

(b) Make a suitable graph of the number of completed passes, and explain briefly why you would have some doubts about using t-procedures in this situation.

Solution

Don't be tempted to think *too* hard about the choice of graph (though I talk more about this below). One quantitative variable, so a histogram again. There are only 12 observations, so 5 bins is about as high as you should go:

```
ggplot(ben, aes(x=completed)) + geom_histogram(bins=5)
```

This one shows an outlier: there is one number of completed passes that is noticeably higher than the rest. A normal distribution doesn't have outliers, and so this, coupled with a small sample in which normality is important, means that we should not be using a t-test or confidence interval.

If you chose a different number of bins, you might get a different look. Here's 4 bins:

```
ggplot(ben, aes(x=completed)) + geom_histogram(bins=4)
```

That looks more like right-skewness, but the conclusion is the same.

Extra: if you have read, for example, Problem 6.1 in PASIAS, you'll have seen that another possibility is a one-group boxplot. This might have been the context in which you first saw the boxplot, maybe at about the time you first saw the five-number summary, but I don't talk about that so much in this course because `ggplot` boxplots have both an `x` and a `y`, and it makes more sense to think about using boxplots to *compare* groups. But, you can certainly get R to make you a one-sample boxplot. What you do is to set the grouping variable to a "dummy" thing like the number 1:

```
ggplot(ben, aes(x=1, y=completed)) + geom_boxplot()
```

and then you ignore the x-axis.

This really shows off the outlier; it is actually *much* bigger than the other observations. It didn't show up so much on the histograms because of where the bin boundaries happened to come. On the four-bin histogram, the highest value 30 was in the 27.5–32.5 bin, and the second-highest value 23 was at the bottom of the 22.5–27.5 bin. So the highest and second-highest values looked closer together than they actually were.

If you have been reading ahead, you might also be thinking about a normal quantile plot. That is for specifically assessing normality, and here this is something that interests us, because a t-test will be doubtful if the normality fails:

```
ggplot(ben, aes(sample=completed)) + stat_qq() + stat_qq_line()
```

This again shows off the outlier at the high end. It is a reasonable choice of plot *here* because normality is of specific interest to us.

A note: you are absolutely not required to read ahead to future lectures. Each assignment can be done using the material in the indicated lectures only. If you want to use something from future lectures, go ahead, but make sure you are using it appropriately.

Don't be tempted to plot the number of completed passes against something like week number:

```
ggplot(ben, aes(x=week, y=completed)) + geom_point()
```

That is quite interesting (a mostly increasing trend over weeks, with the outlier performance in week 10), but it doesn't tell us what we want to know *here*: namely, is a t-test any good?

■

(c) Run a sign test to compare Roethlisberger's performance in 2010 with his previous average of 22.5 completions per game. What do you conclude?

Solution

Use `smmr`, dataframe, column, null median:

```
sign_test(ben, completed, 22.5)
```

```
$above_below
below above
   10     2

$p_values
  alternative    p_value
1       lower 0.01928711
2       upper 0.99682617
3   two-sided 0.03857422
```

I am looking for *any* change, so for me, a two-sided test is appropriate. If you think this is one-sided, make a case for your side, and then go ahead.

My P-value is 0.039, so I can reject the null hypothesis (that the median number of passes completed is 22.5) and conclude that it has changed in 2010.

(You might hypothesize that this is the result of a decrease in confidence, that he is either throwing fewer passes, or the ones that he is throwing are harder to catch. If you know about football, you might suspect that Roethlisberger was actually passing *too much*, including in situations where he should have handing off to the running back, instead of reading the game appropriately.)

Extra: I said above that I cheated and made the null median 22.5 instead of 22. What happens if we make the null median 22?

```
sign_test(ben, completed, 22)
```

```
$above_below
below above
    8     2

$p_values
  alternative   p_value
1       lower 0.0546875
2       upper 0.9892578
3   two-sided 0.1093750
```

For one thing, the result is no longer significant. But looking at the table of values above and below reveals something odd: there are only ten values. What happened to the other two? What happened is that two of the data values were exactly equal to 22, so they are neither above nor below. In the sign test, they are thrown away, so that we are left with 8 values below 22 and 2 above.

I didn't want to make you wonder what happened, so I made the null median 22.5.

■

(d) Why might you have expected your sign test to come out significant, even without looking at the P-value? Explain briefly.

Solution

The other ingredient to the sign test is how many data values are above and below the null median. You can look at the output from `sign_test` (the first part), or count them yourself:

```r
ben %>% count(completed<22.5)
```

```
# A tibble: 2 x 2
  `completed < 22.5`     n
  <lgl>              <int>
1 FALSE                  2
2 TRUE                  10
```

You can put a logical condition (something that can be true or false) into `count`, or you can create a new column using `ifelse` (which I think I showed you somewhere):

```r
ben %>% mutate(side = ifelse(completed<22.5, "below", "above")) %>%
count(side)
```

```
# A tibble: 2 x 2
  side      n
  <chr> <int>
1 above     2
2 below    10
```

Whichever way you do it, there seem to be a lot more values below than above, very different from a 50–50 split. Even with only 12 observations, this turns out to be enough to be significant. (If you tossed a fair coin 12 times, would you be surprised to get only 2 heads or 2 tails?)

■

(e) Obtain a 90% confidence interval for the median number of completed passes (over "all possible games played by 2010 Ben Roethlisberger").

Solution

This is `ci_median`, but with `conf.level` since you are not using the default level of 95%:

```r
ci_median(ben, completed, conf.level = 0.90)
```

```
[1] 17.00244 21.99878
```

17 to 22 completed passes.

Extra: the P-value of the sign test only changes (as the null median changes) when you get to a data point; otherwise, the number of values above and below will stay the same, and the

P-value will stay the same. The data values here were all whole numbers, so the limits of the confidence interval are also whole numbers (to the accuracy of the bisection), so the interval really should be rounded off.

■

(f) Find a 90% confidence interval for the *mean* number of passes completed, and explain briefly why it differs from the one for the median in the way that it does.

Solution

All right, get the interval for the mean first:

```
with(ben, t.test(completed, conf.level = 0.90))
```

```
    One Sample t-test

data:  completed
t = 17.033, df = 11, p-value = 2.971e-09
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 17.89124 22.10876
sample estimates:
mean of x
       20
```

The 95% confidence interval for the mean goes from 17.9 to 22.1 (completions per game).

This is higher at both ends than the interval for the median, though possibly not as much as I expected. This is because the mean is made higher by the outlier (compared to the median), and so the CI procedure comes to the conclusion that the mean is higher.

Extra: this is one of those cases where the bootstrap might shed some light on the sampling distribution of the sample mean:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(ben$completed, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 10)
```

This is noticeably skewed to the right (it goes further up from the peak than down), which is why the CI for the mean went up a bit higher than the one for the median.

Finally, bootstrapping the median is not something you'd want to do, since the sign test doesn't depend on anything being normally-distributed. This is a good thing, since bootstrapping the sample median is weird:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(ben$completed, replace = TRUE))) %>%
  mutate(my_median = median(my_sample)) %>%
  ggplot(aes(x = my_median)) + geom_histogram(bins = 30)
```

The "holes" in the distribution comes about because there are not all that many different possible sample medians when you sample with replacement. For one thing, the values are all whole numbers, so the median can only be something or something and a half. Even then, the bar heights look kind of irregular.

I used a large number of bins to emphasize this, but even a more reasonable number looks strange:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(ben$completed, replace = TRUE))) %>%
  mutate(my_median = median(my_sample)) %>%
  ggplot(aes(x = my_median)) + geom_histogram(bins = 10)
```

A sample median of 19 or 20 is more likely than one of 19.5.

■

7.10 Six ounces of protein

A company produces prepackaged diet meals. These meals are advertised as containing "6 ounces of protein per package". A consumer organization is concerned that this is not accurate. The organization takes a random sample of 20 of these meals, and measures the protein content of each one. The data are in http://ritsokiguess.site/datafiles/protein.txt as one column.

(a) Read in and display (some of) the data.

Solution

The usual. This is one column only, so you can pretend the columns are separated by anything at all and it will still work:

```
my_url <- "http://ritsokiguess.site/datafiles/protein.txt"
meals <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  protein = col_double()
)
```

```r
meals %>% arrange(protein)
```

```
# A tibble: 20 x 1
   protein
     <dbl>
 1     3.6
 2     4.2
 3     4.3
 4     4.4
 5     4.7
 6     4.9
 7     5.1
 8     5.1
 9     5.2
10     5.5
11     5.5
12     5.6
13     5.7
14     5.8
15     5.8
16     6.1
17     6.1
18     6.1
19     6.1
20     6.1
```

Get it to work via one of the methods you've seen in this class (ie., *not* `read.table`); I don't mind how you manage it.

■

(b) Make a suitable graph of your data.

Solution

One quantitative variable, so a histogram with a sufficiently small number of bins:

```r
ggplot(meals, aes(x=protein)) + geom_histogram(bins = 5)
```

■

(c) Why might a sign test be better than a t-test for assessing the average amount of protein per package? Explain briefly. ("Average" here means any measure of centre.)

Solution

The shape of the above distribution is skewed to the left, and not symmetric like a normal distribution. (If you say "the histogram is not normal", make sure you also say how you know.) This means that the median would be a better measure of "average" (that is, centre) than the mean is, because the mean would be pulled downwards by the long tail, and the median would not. To complete the story, the sign test is a test of the median, so the sign test would be better than the t-test, which is a test of the mean.

The other thing you might consider is the sample size, 20, which *might* be large enough to overcome this amount of skewness, but then again it might not be. So you could say that we should be cautious and run the sign test here instead.

■

(d) Run a suitable sign test for these data. What do you conclude?

Solution

First, if you have not already done so, install `smmr` *following the instructions in the lecture notes.* (This one is not just `install.packages`.) Then, make sure you have a `library(smmr)` somewhere above where you are going to use something from it. Once that is in place, remember what we were interested in: was the median protein content 6 ounces, or is there evidence that it is something different? (The "not accurate" in the question says that the median could be higher or lower, either of which would be a problem, and so we need a two-sided alternative.) Thus, the null median is 6 and we need a two-sided test, which goes this way:

```
sign_test(meals, protein, 6)
```

```
$above_below
below above
   15     5

$p_values
  alternative    p_value
1       lower 0.02069473
2       upper 0.99409103
3   two-sided 0.04138947
```

The P-value, 0.0414, is less than 0.05, so we reject the null hypothesis and conclude that the median is different from 6 ounces. The advertisement by the company is not accurate.

Make sure you give the actual P-value you are comparing with 0.05, since otherwise your answer is incomplete. That is, you need to say more than just "the P-value is less than 0.05"; there are three P-values here, and only one of them is the right one.

Extra: we already decided that a t-test is not the best here, but I am curious as to how different its P-value is:

```
with(meals, t.test(protein, mu=6))
```

```
    One Sample t-test

data:  protein
t = -4.2312, df = 19, p-value = 0.000452
alternative hypothesis: true mean is not equal to 6
95 percent confidence interval:
 4.946263 5.643737
```

```
sample estimates:
mean of x
    5.295
```

The conclusion is the same, but the P-value is a lot smaller. I don't think it should really be this small; this is probably because the mean is pulled down by the left skew and so really ought not to look so far below 6. I am inclined to think that if the t-test were correct, its P-value ought to be between this and the one from the sign test, because the t-test uses the actual data values, and the sign test uses the data less efficiently (only considering whether each one is above or below the null median).

■

(e) In your sign test, how could you have deduced that the P-value was going to be small even without looking at any of the P-values themselves? Explain briefly.

Solution

Look at the other part of the output, the count of values above and below the null median. (You might have to click on "R Console" to see it.) If the null hypothesis was correct and the median was really 6, you'd expect to see about half the data values above 6 and about half below. But that is not what happened: there were 15 values below and only 5 above. Such an uneven split is rather unlikely if the null hypothesis was correct. So we would guess that our P-value would be small, as indeed it is.

■

(f) Obtain a 90% confidence interval for the population median protein content. What does this tell you about the reason for the rejection or non-rejection of the null hypothesis above?

Solution

This is `ci_median`, but you need to be paying attention: it's not the default 95% confidence level, so you have to specify that as well:

```r
ci_median(meals, protein, conf.level = 0.90)
```

```
[1] 4.905273 5.793750
```

The interval goes from 4.91 to 5.79. (The data values have one decimal place, so you could justify two decimals in the CI for the median, but anything beyond that is noise and you shouldn't give it in your answer.[5])

This interval is entirely below 6 (the null median), so evidently the reason that we rejected 6 as the population median is that the actual population median is *less than* 6.

Extra: the CI for the median is not that different from the one for the mean, which suggests that maybe the t-test was not so bad after all. If you want to investigate further, you can try finding a bootstrapped sampling distribution of the sample mean, and see how non-normal it looks:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(meals$protein, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) -> d
ggplot(d, aes(x = my_mean)) + geom_histogram(bins = 10)
```

That is pretty close to normal. So the t-test would in actual fact have been fine. To confirm, a normal quantile plot of the bootstrapped sampling distribution:

[5]There is actually slightly more to it here: the ends of this confidence interval for the median are always data values, because of the way it is constructed, so the actual end points really ought to be given to the *same* number of decimals as the data, here 4.9 to 5.8. The output given is not exactly 4.9 and 5.8 because of inaccuracy in the bisection.

```r
ggplot(d, aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```

A *tiny bit* skewed to the left.

But I didn't ask you to do this, because I wanted to give you a chance to do a sign test for what seemed like a good reason.

Extra 2: I mentioned in an note that the endpoints of the CI for the median are actually data points, only we didn't see it because of the accuracy to which `ci_median` was working. You can control this accuracy by an extra input `tol`. Let's do something silly here:

```r
ci_median(meals, protein, conf.level = 0.90, tol = 0.00000001)
```

```
[1] 4.900001 5.799999
```

This takes a bit longer to run, since it has to get the answer more accurately, but now you can see how the interval goes from "just over 4.9" to "just under 5.8", and it actually makes the most sense to give the interval as "4.9 to 5.8" without giving any more decimals.

Extra 3: the reason for the confidence interval endpoints to be data values is that the interval comes from inverting the test: that is, finding the values of the population median that would not be rejected by a sign test run on our data. Recall how the sign test works: it is based on a count of how many data values are above and below the hypothesized population median.

These counts are only going to change as the hypothesized median changes if you hit a data point, since that's the only way you can change how many values are above and below.[6] Thus, the only places where changing the null median changes whether or not a value for it is inside or outside the confidence interval are at data values, and thus the ends of the interval must be at (or, perhaps more strictly, just above or below) data values.

This is a peculiarity of using the sign test to get a CI for the median. If, say, you were to invert the t-test to get a confidence interval for the mean, you wouldn't see that. (This is in fact exactly what you do to get a confidence interval for the mean, but this is not the way it is usually introduced.) The reason that the CI for the mean (based on the t-test) is different from the one for the median (based on the sign test) is that if you change the null hypothesis in the t-test, however slightly, you change the P-value (maybe only slightly, but you change it). So the CI for the mean, based on the t-test, is not required to have data points at its ends, and indeed usually does not. The difference is in the kind of distribution the test statistic has; the t-distribution is continuous, while the sign test statistic (a count of the number of values above or below something) is discrete. It's the discreteness that causes the problems.

■

[6] There is a technicality about what happens when the null median is exactly equal to a data value; see PASIAS for more discussion on this.

8 Mood median test

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   0.5.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(smmr)
```

8.1 Sugar in breakfast cereals

The data for this question are in http://ritsokiguess.site/datafiles/cereal-sugar.txt. The story here is whether breakfast cereals marketed to children have a lot of sugar in them; in particular, whether they have more sugar on average than cereals marketed to adults.

(a) Read in the data (to R) and display the data set. Do you have a variable that distinguishes the children's cereals from the adults' cereals, and another that contains the amount of sugar?

(b) Calculate the mean sugar content for each group of cereals (the adults' ones and the children's ones). Do they look similar or different?

(c) Make side-by-side boxplots of the sugar contents of the two types of cereal. What do you see that is out of the ordinary?

(d) Explain briefly why you would not trust a two-sample t-test with these data. (That is, say what the problem is, and why it's a problem.)

(e) Run a suitable test to see whether the "typical" amount of sugar differs between adult's and children's cereals. Justify the test that you run. (You can use the version of your test that lives in a package, if that is easier for you.) What do you conclude, in the context of the data?

8.2 Fear of math

Two new short courses have been proposed for helping students who suffer from severe math phobia. The courses are labelled A and B. Ten students were randomly allocated to one of these two courses, and each student's score on a math phobia test was recorded after they completed their course. The math phobia test produces whole-number scores between 0 and 10, with a higher score indicating a greater fear of mathematics. The data can be found in link. We start with R for this question.

(a) Read in the data and check, however you like, that you have 10 observations, 5 from each course.

(b) Do a two-sample t-test to assess whether there is a difference in mean phobia scores after the students have taken the two courses. What do you conclude? (You have no *a priori*[1] reason to suppose that a particular one of the tests will produce a higher mean than the other, so do a two-sided test.)

(c) Draw boxplots of the math phobia scores for each group (one line of code). What is the most striking thing that you notice?

(d) Explain briefly why a t-test would not be good for these data. (There are two things that you need to say.)

(e) Run a suitable test to compare the "typical" scores for the two courses. (You can use the version from a package rather than building your own.) What do you conclude?

8.3 Medical instructions

Do people understand medical instructions better at certain times of the day? In a study, students in a grade 12 class are randomly divided into two groups, A and B. All students see a video describing how to use an infant forehead thermometer. The students in Group A see the video at 8:30 am, while the students in Group B see the same video at 3:00 pm (on the same day). The next day, all the students are given a test on the material in the video (graded out of 100). The observed scores are in link (values separated by spaces).

[1] That is, before looking at the data. This is Latin. It's also the place that the Bayesian "prior distribution" comes from. The "posterior distribution" comes from the Latin *a posteriori*, which means *afterwards*, that is, after you have looked at the data.

(a) Read the data into R and display the (first ten) values.

(b) Obtain a suitable plot that will enable you to assess the assumptions for a two-sample t-test.

(c) Why might you have doubts about using a two-sample t-test here?

(d) Run Mood's median test as in class (*without* using `smmr`). What do you conclude, in the context of the data? What recommendation would you make about the time of day to see the video? (You might get a warning about "chisquared approximation being incorrect", which you can ignore here.)

(e) Run Mood's median test on these data using my `smmr` package, and verify that you get the same answer.

8.4 Handspans revisited

Take your right hand, and stretch the fingers out as far as you can. The distance between the tip of your thumb and the tip of your little (pinky) finger is your handspan. The students in a Statistics class at Penn State measured their handspans and also whether they identified as male or female. The data are at http://ritsokiguess.site/datafiles/handspan.txt, with handspans measured in inches. We want to see whether male students have a larger mean handspan than female students.

(a) Read in and display (some of) the data.

(b) Make a suitable (facetted) normal quantile plot of the data. (Bear in mind what is supposed to have a normal distribution.)

(c) Discuss briefly whether you might prefer to use Mood's median test to compare the handspans of the male and female students, compared to a two-sample t-test.

(d) Run Mood's median test. What do you conclude from the test, in the context of the data?

My solutions follow:

8.5 Sugar in breakfast cereals

The data for this question are in http://ritsokiguess.site/datafiles/cereal-sugar.txt. The story here is whether breakfast cereals marketed to children have a lot of sugar in them; in particular, whether they have more sugar on average than cereals marketed to adults.

(a) Read in the data (to R) and display the data set. Do you have a variable that distinguishes the children's cereals from the adults' cereals, and another that contains the amount of sugar?

Solution

```
my_url="http://ritsokiguess.site/datafiles/cereal-sugar.txt"
cereals=read_delim(my_url," ")
```

```
Rows: 40 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): who
dbl (1): sugar

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
cereals
```

```
# A tibble: 40 x 2
   who      sugar
   <chr>    <dbl>
 1 children  40.3
 2 children  55
 3 children  45.7
 4 children  43.3
 5 children  50.3
 6 children  45.9
 7 children  53.5
 8 children  43
 9 children  44.2
10 children  44
# i 30 more rows
```

The variable `who` is a categorical variable saying who the cereal is intended for, and the variable `sugar` says how much sugar each cereal has.

■

(b) Calculate the mean sugar content for each group of cereals (the adults' ones and the children's ones). Do they look similar or different?

Solution

`group_by` and `summarize`:

```
cereals %>% group_by(who) %>%
summarize(sugar_mean=mean(sugar))
```

```
# A tibble: 2 x 2
  who      sugar_mean
  <chr>         <dbl>
1 adults         10.9
2 children       46.6
```

These means look very different, though it would be better to look at a boxplot (coming up in a moment).

■

(c) Make side-by-side boxplots of the sugar contents of the two types of cereal. What do you see that is out of the ordinary?

Solution

The usual:

```
ggplot(cereals,aes(x=who,y=sugar))+geom_boxplot()
```

I see outliers: two high ones on the adults' cereals, and one high and one low on the children's cereals.

My thought above about the means being very different is definitely supported by the medians being very different on the boxplots. We should have no trouble declaring that the "typical" amounts of sugar in the adults' and children's cereals are different.

■

(d) Explain briefly why you would not trust a two-sample t-test with these data. (That is, say what the problem is, and why it's a problem.)

Solution

The problem is the outliers (which is rather a giveaway), but the reason it's a problem is that the two-sample t-test assumes (approximately) normal data, and a normal distribution doesn't have outliers. Not only do you need to note the outliers, but you also need to say why the outliers cause a problem *in this case*. Anything less than that is not a complete answer.

■

(e) Run a suitable test to see whether the "typical" amount of sugar differs between adult's and children's cereals. Justify the test that you run. (You can use the version of your test that lives in a package, if that is easier for you.) What do you conclude, in the context of the data?

Solution

Having ruled out the two-sample t-test, we are left with Mood's median test. I didn't need you to build it yourself, so you can use package `smmr` to run it with:

```
library(smmr)
median_test(cereals,sugar,who)
```

```
$table
          above
group      above below
  adults       2    19
  children    18     1

$test
       what        value
1 statistic 2.897243e+01
2        df 1.000000e+00
3   P-value 7.341573e-08
```

We conclude that there *is* a difference between the median amounts of sugar between the two groups of cereals, the P-value of 0.00000007 being extremely small.

Why did it come out so small? Because the amount of sugar was smaller than the overall median for almost all the adult cereals, and larger than the overall median for almost all the children's ones. That is, the children's cereals really do have more sugar.

Mood's median test doesn't come with a confidence interval (for the difference in population medians), because whether or not a certain difference in medians is rejected depends on what those medians actually are, and the idea of the duality of the test and CI doesn't carry over as we would like.

My daughter likes chocolate Cheerios, but she also likes Shredded Wheat and *Bran Flakes*. Go figure. (Her current favourite is Raisin Bran, even though she doesn't like raisins by themselves.)

Mood's median test is the test we should trust, but you might be curious about how the t-test stacks up here:

```
t.test(sugar~who,data=cereals)
```

```
    Welch Two Sample t-test
```

```
data:  sugar by who
t = -11.002, df = 37.968, p-value = 2.278e-13
alternative hypothesis: true difference in means between group adults and group children is n
95 percent confidence interval:
 -42.28180 -29.13925
sample estimates:
  mean in group adults mean in group children
              10.90000               46.61053
```

The P-value is *even smaller*, and we have the advantage of getting a confidence interval for the difference in means: from about 30 to about 40 units less sugar in the adult cereals. Whatever the units were.

■

8.6 Fear of math

Two new short courses have been proposed for helping students who suffer from severe math phobia. The courses are labelled A and B. Ten students were randomly allocated to one of these two courses, and each student's score on a math phobia test was recorded after they completed their course. The math phobia test produces whole-number scores between 0 and 10, with a higher score indicating a greater fear of mathematics. The data can be found in link. We start with R for this question.

(a) Read in the data and check, however you like, that you have 10 observations, 5 from each course.

Solution

This doesn't need much comment:

```r
my_url <- "http://ritsokiguess.site/datafiles/mathphobia.txt"
math <- read_delim(my_url, " ")
```

```
Rows: 10 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): course
dbl (1): phobia

i Use `spec()` to retrieve the full column specification for this data.
```

```
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
math
```

```
# A tibble: 10 x 2
   course phobia
   <chr>   <dbl>
 1 a           8
 2 a           7
 3 a           7
 4 a           6
 5 a           6
 6 b           9
 7 b           8
 8 b           7
 9 b           2
10 b           1
```

This will do, counting the `a` and `b`. Or, to save yourself that trouble:

```
math %>% count(course)
```

```
# A tibble: 2 x 2
  course     n
  <chr>  <int>
1 a          5
2 b          5
```

Five each. The story is to get the computer to do the grunt work for you, if you can make it do so. Other ways:

```
math %>% group_by(course) %>% summarize(count = n())
```

```
# A tibble: 2 x 2
  course count
  <chr>  <int>
1 a          5
2 b          5
```

and this:

```r
with(math, table(course))
```

```
course
a b
5 5
```

giving the same answer. Lots of ways.

Extra: there is an experimental design issue here. You might have noticed that each student did only *one* of the courses. Couldn't students do both, in a matched-pairs kind of way? Well, it's a bit like the kids learning to read in that if the first of the courses reduces a student's anxiety, the second course won't appear to do much good (even if it actually would have been helpful had the student done that one first). This is the same idea as the kids learning to read: once you've learned to read, you've learned to read, and learning to read a second way won't help much. The place where matched pairs scores is when you can "wipe out" the effect of one treatment before a subject gets the other one. We have an example of kids throwing baseballs and softballs that is like that: if you throw one kind of ball, that won't affect how far you can throw the other kind.

■

(b) Do a two-sample t-test to assess whether there is a difference in mean phobia scores after the students have taken the two courses. What do you conclude? (You have no *a priori*[2] reason to suppose that a particular one of the tests will produce a higher mean than the other, so do a two-sided test.)

Solution

A two-sided test is the default, so there is not much to do here:

```r
t.test(phobia ~ course, data = math)
```

```
    Welch Two Sample t-test

data:  phobia by course
t = 0.83666, df = 4.4199, p-value = 0.4456
alternative hypothesis: true difference in means between group a and group b is not equal to
95 percent confidence interval:
```

[2]That is, before looking at the data. This is Latin. It's also the place that the Bayesian "prior distribution" comes from. The "posterior distribution" comes from the Latin *a posteriori*, which means *afterwards*, that is, after you have looked at the data.

```
 -3.076889  5.876889
sample estimates:
mean in group a mean in group b
            6.8             5.4
```

The P-value of 0.4456 is nowhere near less than 0.05, so there is no evidence at all that the mean math phobia scores are different between the two courses.

■

(c) Draw boxplots of the math phobia scores for each group (one line of code). What is the most striking thing that you notice?

Solution

```
ggplot(math, aes(x = course, y = phobia)) + geom_boxplot()
```

Boxplot `a` is just weird. The bar across the middle is actually at the top, and it has no bottom. (Noting something sensible like this is enough.) Boxplot `b` is hugely spread out.[3]

[3]The two groups have very different spreads, but that is not a problem as long as we remember to do the Welch-Satterthwaite test that does not assume equal spreads. This is the default in R, so we are good, at least with that.

By way of explanation: the course `a` scores have a number of values equal so that the 3rd quartile and the median are the name, and also that the first quartile and the minimum value are the same:

```
tmp <- math %>% filter(course == "a")
tmp %>% count(phobia)
```

```
# A tibble: 3 x 2
  phobia     n
   <dbl> <int>
1      6     2
2      7     2
3      8     1
```

```
summary(tmp)
```

```
    course              phobia
 Length:5           Min.   :6.0
 Class :character   1st Qu.:6.0
 Mode  :character   Median :7.0
                    Mean   :6.8
                    3rd Qu.:7.0
                    Max.   :8.0
```

The phobia scores from course A are two 6's, two 7's and an 8. The median and third quartile are both 7, and the first quartile is the same as the lowest value, 6.

Technique note: I wanted to do two things with the phobia scores from course A: count up how many of each score, and show you what the five-number summary looks like. One pipe won't do this (the pipe "branches"), so I saved what I needed to use, before it branched, into a data frame `tmp` and then used `tmp` twice. Pipes are powerful, but not *all*-powerful.

■

(d) Explain briefly why a t-test would not be good for these data. (There are two things that you need to say.)

Solution

The easiest way to structure this is to ask yourself first what the t-test needs, and second whether you have it. The t-test assumes (approximately) normal data. The boxplot for group `a` doesn't even look symmetric, and the one for group `b` has an oddly asymmetric box. So I

think the normality is in question here, and therefore another test would be better. (This is perhaps a bit glib of an answer, since there are only 5 values in each group, and so they can certainly look non-normal even if they actually are normal, but these values are all integers, so it is perhaps wise to be cautious.) We have the machinery to assess the normality for these, in one shot:

```
ggplot(math, aes(sample = phobia)) +
  stat_qq() + stat_qq_line() +
  facet_wrap(~course, ncol = 1, scales = "free")
```

I don't know what *you* make of those, but they both look pretty straight to me (and there are only five observations, so it's hard to judge). Course `b` maybe has a "hole" in it (three large values and two small ones). Maybe. I dunno. What I would *really* be worried about is outliers, and at least we don't have those. I mentioned in class that the t-tests are robust to non-normality. I ought to have expanded on that a bit: what really makes the t-test still behave itself with non-normality is when you have *large* samples, that is, when the Central Limit Theorem has had a chance to take hold. (That's what drives the normality not really being necessary in most cases.) But, even with small samples, exact normality doesn't matter so much. Here, we have two tiny samples, and so we have to insist a bit more, but only a bit more, on a more-or-less normal shape in each group. (It's kind of a double jeopardy in that the situation where normality matters most, namely with small samples, is where it's the hardest to judge, because samples of size 5 even from a normal distribution can look very non-normal.) But, the biggest threats to the t-test are big-time skewness and outliers, and we

are not suffering too badly from those.

■

(e) Run a suitable test to compare the "typical" scores for the two courses. (You can use the version from a package rather than building your own.) What do you conclude?

Solution

This is an invite to use `smmr`:

```
library(smmr)
median_test(math, phobia, course)
```

```
$table
     above
group above below
    a     1     2
    b     2     2

$test
       what     value
1 statistic 0.1944444
2        df 1.0000000
3   P-value 0.6592430
```

We are nowhere near rejecting equal medians; in fact, both courses are very close to 50–50 above and below the overall median.

If you look at the frequency table, you might be confused by something: there were 10 observations, but there are only $1 + 2 + 2 + 2 = 7$ in the table. This is because three of the observations were equal to the overall median, and had to be thrown away:

```
math %>% summarize(med = median(phobia))
```

```
# A tibble: 1 x 1
    med
  <dbl>
1     7
```

```
math %>% count(phobia)
```

```
# A tibble: 6 x 2
  phobia     n
   <dbl> <int>
1      1     1
2      2     1
3      6     2
4      7     3
5      8     2
6      9     1
```

The overall median was 7. Because the actual data were really discrete (the phobia scores could only be whole numbers), we risked losing a lot of our data when we did this test (and we didn't have much to begin with). The other thing to say is that with small sample sizes, the frequencies in the table have to be *very* lopsided for you to have a chance of rejecting the null. Something like this is what you'd need:

```r
x <- c(1, 1, 2, 6, 6, 6, 7, 8, 9, 10)
g <- c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2)
d <- tibble(x, g)
median_test(d, x, g)
```

```
$table
     above
group above below
    1     0     3
    2     4     0

$test
       what       value
1 statistic 7.000000000
2        df 1.000000000
3   P-value 0.008150972
```

I faked it up so that we had 10 observations, three of which were equal to the overall median. Of the rest, all the small ones were in group 1 and all the large ones were in group 2. This is lopsided enough to reject with, though, because of the small frequencies, there actually was a warning about "chi-squared approximation may be inaccurate".[4]

■

[4]There *was*, in the `chisq.test` inside `median_test`, but in `smmr` I didn't pass that warning back to the outside world.

8.7 Medical instructions

Do people understand medical instructions better at certain times of the day? In a study, students in a grade 12 class are randomly divided into two groups, A and B. All students see a video describing how to use an infant forehead thermometer. The students in Group A see the video at 8:30 am, while the students in Group B see the same video at 3:00 pm (on the same day). The next day, all the students are given a test on the material in the video (graded out of 100). The observed scores are in link (values separated by spaces).

(a) Read the data into R and display the (first ten) values.

Solution

Separated by spaces, so `read_delim`:

```
my_url <- "http://ritsokiguess.site/datafiles/forehead.txt"
instr <- read_delim(my_url, " ")
```

```
Rows: 18 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): group
dbl (1): score

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
instr
```

```
# A tibble: 18 x 2
   group score
   <chr> <dbl>
 1 A        88
 2 A        89
 3 A        79
 4 A       100
 5 A        98
 6 A        89
 7 A        65
 8 A        94
 9 A        95
```

```
10 A         91
11 B         87
12 B         69
13 B         78
14 B         79
15 B         83
16 B         90
17 B         85
18 B         58
```

■

(b) Obtain a suitable plot that will enable you to assess the assumptions for a two-sample t-test.

Solution

We need the values in each group to be approximately normally distributed. Side-by-side boxplots will do it:

```
ggplot(instr, aes(x = group, y = score)) + geom_boxplot()
```

or, if you like, separate (facetted) normal quantile plots, which I would do this way:

```r
ggplot(instr, aes(sample = score)) +
  stat_qq() + stat_qq_line() +
  facet_wrap(~group, ncol = 1)
```

■

(c) Why might you have doubts about using a two-sample t-test here?

Solution

We are looking for non-normality in at least one of the groups. Here, both groups have an outlier at the low end that would be expected to pull the mean downward. I don't think there is left-skewness here, since there is no particular evidence of the high-end values being bunched up: the problem in both cases with normality is at the low end. One way or another, I'm expecting you to have noticed the outliers. Extra: last year, when I first drew the normal quantile plots, there was no `stat_qq_line`, so you had to imagine where the line went if you did it this way. Without the line, these plots look somewhat curved, which would have pointed to left-skewness, but now we see that the lowest observation is too low, and maybe the second-lowest one as well, while the other observations are just fine.

■

(d) Run Mood's median test as in class (*without* using `smmr`). What do you conclude, in the context of the data? What recommendation would you make about the time of day

to see the video? (You might get a warning about "chisquared approximation being incorrect", which you can ignore here.)

Solution

The overall median first:

```
instr %>% summarize(med = median(score))
```

```
# A tibble: 1 x 1
    med
  <dbl>
1  87.5
```

87.5, which is not equal to any of the data values (they are all integers). This will avoid any issues with values-equal-to-median later.

Then, create and save a table of the value by group and above/below median. You can count either above or below (it comes out equivalently either way):

```
tab <- with(instr, table(group, score > 87.5))
tab
```

```
group FALSE TRUE
    A     2    8
    B     7    1
```

Then, chi-squared test for independence (the null) or association of some kind (the alternative). The `correct=F` is saying not to do Yates's correction, so that it would come out the same if you were doing it by hand ("observed minus expected, squared, divided by expected" and all that stuff).

```
chisq.test(tab, correct = F)
```

```
Warning in chisq.test(tab, correct = F): Chi-squared approximation may be
incorrect


    Pearson's Chi-squared test
```

```
data:  tab
X-squared = 8.1, df = 1, p-value = 0.004427
```

The P-value is 0.0044, which is (much) smaller than 0.05, and therefore you can reject independence and conclude association: that is, whether a student scores above or below the median depends on which group they are in, or, that the median scores are different for the two groups.

The warning is because the expected frequencies are on the small side (if you have done this kind of problem by hand, you might remember something about "expected frequencies less than 5". This is that.) Here, the P-value is so small that we can afford to have it be inaccurate by a bit and still not affect the conclusion, so I think we are safe.

As for which group is better, well, the easiest way is to go back to your boxplots and see that the median for group A (8:30 am) is substantially higher than for group B (3:00pm). But you can also see it from your frequency table, if you displayed it:

```
tab
```

```
group FALSE TRUE
    A     2    8
    B     7    1
```

Most of the people in the 8:30 am group scored above the median, and most of the people in the 3:00 pm group scored below the median. So the scores at 8:30 am were better overall.

As I write this, it is just after 3:00 pm and I am about to make myself a pot of tea!

Extra: about that `correct=F` thing. There was a point of view for a long time that when you are dealing with a 2×2 table, you can get better P-values by, before squaring "observed minus expected", taking 0.5 away from the absolute value of the difference. This is called Yates's correction. It is in the same spirit as the "continuity correction" that you might have encountered in the normal approximation to the binomial, where in the binomial you have to have a whole number of successes, but the normal allows fractional values as well. In about the 1960s, the usefulness of Yates's correction was shot down, for general contingency tables. There is, however, one case where it *is* useful, and that is the case where the row totals and column totals are *fixed*.

What do I mean by that? Well, first let's look at a case where the totals are *not* all fixed. Consider a survey in which you want to see whether males and females agree or disagree on some burning issue of the day. You collect random samples of, say, 500 males and 500 females,

and you count how many of them say Yes or No to your statement.[5] You might get results like this:

```
Yes  No  Total
Males    197 303   500
Females  343 157   500
Total    540 460  1000
```

In this table, the row totals must be 500, because you asked this many males and this many females, and each one must have answered something. The column totals, however, are not fixed: you didn't know, ahead of time, that 540 people would answer "yes". That was just the way the data turned out, and if you did another survey with the same design, you'd probably get a different number of people saying "yes".

For another example, let's go back to Fisher (yes, *that* Fisher). A "lady" of his acquaintance claimed to be able, by drinking a cup of tea with milk and sugar in it, whether the milk or the sugar had been added first. Fisher, or, more likely, his housekeeper, prepared 8 cups of tea, 4 with milk first and 4 with sugar first. The lady knew that four of the cups had milk first, and her job was to say which four. The results might have been like this:

```
Actual
Milk first  sugar first  Total
Lady   Milk first        3            1         4
says   sugar first       1            3         4
Total             4            4         8
```

This time, all of the row totals and all of the column totals must be 4, regardless of what the lady thinks. Even if she thinks 5 of the cups of tea actually had milk first, she is going to pick 4 of them to say that they have milk first, since she knows there are only 4. In this case, all of the row and column totals are fixed at 4, and the right analysis is called Fisher's Exact Test, based on the hypergeometric distribution. In a 2×2 table like this one, there is only one "degree of freedom", since as soon as you specify one of the frequencies, say the number of cups where the lady said milk first and they actually were milk first, you can work out the others. But, leaving that aside, the usual chi-squared analysis is a perfectly good approximation, especially if the frequencies are large, and especially if you use Yates's correction.

It is clear that Fisher must have been English, since he was able to get a publication out of drinking tea.

[5] To simplify things, we'll assume that everyone gave a Yes or a No answer, though you could add a column like "No answer" if you wanted to make it more realistic.

How does that apply to Mood's median test? Well, let's remind ourselves of the table we had:

```
tab
```

```
group FALSE TRUE
    A     2    8
    B     7    1
```

We know how many students were in each group: 10 in group A and 8 in B. So the row totals are fixed. What about the columns? These are whether each observation was above or below the overall median. There were 18 observations altogether, so there *must* be 9 above and 9 below.[6] So the column totals are fixed as well. All totals fixed, so we should be using Yates's correction. I didn't, because I wanted to keep things simple, but I should have done.

R's `chisq.test` by default *always* uses Yates's correction, and if you don't want it, you have to say `correct=F`. Which is why I have been doing so all through.

■

(e) Run Mood's median test on these data using my `smmr` package, and verify that you get the same answer.

Solution

Not much to it, since the data is already read in:

```
library(smmr)
median_test(instr, score, group)
```

```
$table
     above
group above below
    A     8     2
    B     1     7

$test
       what       value
1 statistic 8.100000000
2        df 1.000000000
3   P-value 0.004426526
```

[6]Except in the case of the previous problem, where there were multiple observations equal to the overall median. Which we ignore for the moment.

Identical, test statistic, degrees of freedom and P-value. The table of frequencies is also the same, just with columns rearranged. (In `smmr` I counted the number of values below the overall median, whereas in my build-it-yourself I counted the number of values above.)

■

8.8 Handspans revisited

Take your right hand, and stretch the fingers out as far as you can. The distance between the tip of your thumb and the tip of your little (pinky) finger is your handspan. The students in a Statistics class at Penn State measured their handspans and also whether they identified as male or female. The data are at http://ritsokiguess.site/datafiles/handspan.txt, with handspans measured in inches. We want to see whether male students have a larger mean handspan than female students.

(a) Read in and display (some of) the data.

Solution

Delimited by a single space, so:

```
my_url <- "http://ritsokiguess.site/datafiles/handspan.txt"
span <- read_delim(my_url, " ")
```

```
Rows: 190 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): sex
dbl (1): handspan

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
span
```

```
# A tibble: 190 x 2
   sex   handspan
   <chr>    <dbl>
 1 M         21.5
 2 M         22.5
 3 M         23.5
```

```
 4 F         20
 5 F         19
 6 F         20.5
 7 F         20.5
 8 F         20.2
 9 M         23
10 M         24.5
# i 180 more rows
```

■

(b) Make a suitable (facetted) normal quantile plot of the data. (Bear in mind what is supposed to have a normal distribution.)

Solution

Here, we need *each* group to be approximately normal, so make normal quantile plots of handspan, facetted by sex:

```
ggplot(span, aes(sample=handspan)) + stat_qq() + stat_qq_line() +
facet_wrap(~sex)
```

■

(c) Discuss briefly whether you might prefer to use Mood's median test to compare the handspans of the male and female students, compared to a two-sample t-test.

Solution

A two-sample t-test assumes that each of the two samples comes from a (approximately) normal distribution ("the data are normal" is not precise enough). The female values, on the left, definitely have some outliers at the low end (or a long lower tail), so these are definitely not normal. The male values (on the right) are slightly skewed to the left, or there are some mild outliers at the low end, or, if you prefer, these are approximately normal. (You need discussion of each of the males and females, or of why looking at one group is enough.) Because the males are not close enough to normal (or, because neither group is close enough to normal), we would prefer to use Mood's median test. (Say this.) You do yourself a favour by making it clear that you know that *both* groups have to be normal enough; if one is good but the other is not, that is not enough.

The other relevant issue is sample size. The best answer discusses that as well, even though you have a lot to think about already. This data set has 190 observations in it, so the samples must be pretty big:

```
span %>% count(sex)
```

```
# A tibble: 2 x 2
  sex       n
  <chr> <int>
1 F       103
2 M        87
```

With these sample sizes, we can expect a lot of help from the central limit theorem. The apparent outliers in the males won't be a problem, and maybe we could even get away with those outliers in the females.

Extra: you could also think about bootstrapped sampling distributions of the sample mean here. The one we are most concerned about is the females; if it turns out that they are all right, then the males must be all right too, since the plot for them is showing less non-normality (or, without the double negative, is closer to being normal). So let's do the females:

```
span %>%
  filter(sex == "F") -> females
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(females$handspan, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
```

```
ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```

My take is that the sampling distribution of the sample mean for the females is normal enough, therefore the one for the males is also normal enough, therefore the two-sample t-test is actually fine.

The reason that this one is close to normal is different from the other one, though. In the other question, we had milder non-normality but a smaller sample; in this one, the data distribution is less normal, but we had a much larger sample size to compensate.

■

(d) Run Mood's median test. What do you conclude from the test, in the context of the data?

Solution

```
library(smmr)
median_test(span, handspan, sex)
```

```
$table
     above
group above below
```

```
    F    17    82
    M    65    11

$test
       what       value
1 statistic 8.06725e+01
2        df 1.00000e+00
3   P-value 2.66404e-19
```

The P-value of 2.66×10^{-19} is extremely small, so we can conclude that males and females have different median handspans. Remember that we are now comparing medians, and that this test is *two*-sided.

You can stop here, or you can go on and note that most of the males have a handspan bigger than the median, and most of the females have a handspan smaller than the median, so that males have on average a larger handspan. But you have to make the case that males have a larger handspan; you cannot just assert this from the P-value.

A more formal way to do this is to make the same observation as above, then note that this is "on the correct side" (for males to have a larger handspan), and thus that you can halve the P-value, and conclude that males' handspans are indeed larger in terms of median.

Extra: you are probably expecting a confidence interval now for the difference in medians. I haven't talked about that in lecture, because the ideas are a bit trickier than they were for the confidence interval for the sign test. The sign test could be used for testing any median, so we could try a bunch of medians and see whether each one was rejected or not. The problem with Mood's median test is that it only tests that the medians are *the same*. If you could easily test that the difference in medians was 3, say, you would know whether 3 was inside or outside the confidence interval for the difference in medians.

What were the actual sample medians, anyway?

```r
span %>% group_by(sex) %>%
summarize(med = median(handspan))
```

```
# A tibble: 2 x 2
  sex     med
  <chr> <dbl>
1 F      20
2 M      22.5
```

Here's an idea: if we shift all the female handspans up by 2.5 inches, the medians would be the same:

```r
span %>% mutate(x = ifelse(sex=="F", handspan+2.5, handspan)) -> d
d
```

```
# A tibble: 190 x 3
   sex   handspan     x
   <chr>    <dbl> <dbl>
 1 M         21.5  21.5
 2 M         22.5  22.5
 3 M         23.5  23.5
 4 F         20    22.5
 5 F         19    21.5
 6 F         20.5  23
 7 F         20.5  23
 8 F         20.2  22.8
 9 M         23    23
10 M         24.5  24.5
# i 180 more rows
```

Dataframe `d` has a new column `x` that is the handspan plus 2.5 inches for females, and the unchanged handspan for males. So the median of `x` should be the same for males and females:

```r
d %>% group_by(sex) %>%
summarize(med_x = median(x))
```

```
# A tibble: 2 x 2
  sex   med_x
  <chr> <dbl>
1 F      22.5
2 M      22.5
```

and also the medians of `x` cannot possibly be significantly different:

```r
median_test(d, x, sex)
```

```
$table
     above
group above below
    F    46    36
    M    41    35
```

```
$test
       what      value
1 statistic 0.07369901
2        df 1.00000000
3   P-value 0.78602526
```

Quite a lot of the values of `x` are exactly equal to the overall median (and are discarded), so the P-value is not exactly 1 as you would expect. But it is definitely not significant, and so a difference of 2.5 inches smaller for females is going to be in a confidence interval for the difference in medians.

The strategy now is to try shifting the female handspans by different amounts, run Mood's median test for each one, and see which shifts are not rejected. These are the ones for which that difference in medians would be in the confidence interval. Before we get to that, though, I want to simplify the procedure we have, so that it is easier to run it lots of times. First, let's get just the P-value out of the median test:

```
d.1 <- median_test(d, x, sex)
d.1 %>% pluck("test", "value", 3)
```

```
[1] 0.7860253
```

That's the P-value. `pluck` pulls individual things out of bigger things. The variable I called `d.1` has two things in it. The one called `table` has the numbers of data values above and below the overall median; the one called `test` has the test statistic and P-value in it. `test` is a dataframe; inside *that* is a column called `what` and a column called `value` with the number we want in it, and we want the third thing in that (the other two are the chi-squared test statistic and its degrees of freedom). Hence the `pluck` statement got the right thing.

Let's think strategy: we want to shift the female handspans by a bunch of different amounts, run the test on each one, and get the P-value each time. When you're running a big for-each like this, you want the thing you do each time to be as simple as possible. So let's write a function that takes the shift as input, works out the new `x`, runs the test, and returns the P-value. We have all the ingredients, so it's a matter of putting them together:

```
shift_pval <- function(shift) {
  span %>% mutate(x = ifelse(sex == "F", handspan + shift, handspan)) -> d
  d.1 <- median_test(d, x, sex)
  d.1 %>% pluck("test", "value", 3)
}
```

In the function, the `shift` is input. The first line computes the handspans shifted by the input amount, whatever it is; the second line runs the median test on the shifted data; the last line pulls out, and returns, the P-value.

I am being a little sloppy here (but R is letting me get away with it): the function is also using a dataframe called `span`, which is the one we read in from the file earlier. That *was not input to the function*, so, if you have experience with other programming languages, you might be wondering whether that is "in the scope" of inside the function: that is, whether R will know about it. R does; anything the function needs that is not part of the input, it will take from your workspace. This is, you might imagine, dangerous; if the input to your function is called, say, `x`, you might easily have an `x` lying around in your workspace from some other analysis that has nothing to do with the `x` you want as the input to your function. The safe way to do it, and what I should have done, is to have `span` be input to my function as well. However, that clutters up the discussion below, so we'll leave things as I did them here.

Let's test this on a shift of 2.5 inches, and on the original data (a shift of zero):

```
shift_pval(2.5)
```

```
[1] 0.7860253
```

```
shift_pval(0)
```

```
[1] 2.66404e-19
```

Those are the same P-values we got before, so good.

Now, let's get a bunch of shifts, say from 0 to 5 in steps of 0.5:

```
tibble(shift = seq(0, 5, 0.5))
```

```
# A tibble: 11 x 1
   shift
   <dbl>
 1   0
 2   0.5
 3   1
 4   1.5
 5   2
 6   2.5
 7   3
```

```
 8   3.5
 9   4
10   4.5
11   5
```

work out the P-value for each one, rowwise:

```
tibble(shift = seq(0, 5, 0.5)) %>%
  rowwise() %>%
  mutate(p_value = shift_pval(shift))
```

```
# A tibble: 11 x 2
# Rowwise:
   shift  p_value
   <dbl>    <dbl>
 1   0   2.66e-19
 2   0.5 7.38e-15
 3   1   6.41e- 9
 4   1.5 5.46e- 5
 5   2   5.29e- 2
 6   2.5 7.86e- 1
 7   3   5.15e- 3
 8   3.5 1.13e- 5
 9   4   1.89e- 9
10   4.5 1.84e-14
11   5   1.61e-18
```

and finally decide whether each shift is inside or outside the CI (because I am too lazy to figure out the scientific notation):

```
tibble(shift = seq(0, 5, 0.5)) %>%
  rowwise() %>%
  mutate(p_value = shift_pval(shift)) %>%
  mutate(where = ifelse(p_value<0.05, "outside", "inside"))
```

```
# A tibble: 11 x 3
# Rowwise:
   shift  p_value where
   <dbl>    <dbl> <chr>
 1   0   2.66e-19 outside
```

```
 2   0.5 7.38e-15 outside
 3   1   6.41e- 9 outside
 4   1.5 5.46e- 5 outside
 5   2   5.29e- 2 inside
 6   2.5 7.86e- 1 inside
 7   3   5.15e- 3 outside
 8   3.5 1.13e- 5 outside
 9   4   1.89e- 9 outside
10   4.5 1.84e-14 outside
11   5   1.61e-18 outside
```

The confidence interval goes from 2 inches to 2.5 inches on this scale. I checked and it goes up to 3 really, except that 3 itself is outside the interval. So let's call it 2 to 3 inches. This means that the median female handspan is between 2 and 3 inches *smaller* than the median male handspan, because we had to shift the female handspans up by that much to make them not significantly different.

You, of course, would do just the last pipeline; I showed you the steps so you could see what was going on.

The final observation is that this interval is a long way from containing zero, because the P-value was so tiny.

Let's see how the t-interval looks in comparison (two-sided, because we want the confidence interval):

```r
t.test(handspan~sex, data = span)
```

```
    Welch Two Sample t-test

data:  handspan by sex
t = -10.871, df = 187.92, p-value < 2.2e-16
alternative hypothesis: true difference in means between group F and group M is not equal to
95 percent confidence interval:
 -3.001496 -2.079466
sample estimates:
mean in group F mean in group M
       20.01699        22.55747
```

Almost exactly the same (except that F is before M). So it made no difference at all whether we did a t-test or a Mood's median test, as the bootstrapped sampling distribution suggested.

■

9 Matched pairs t and sign test

```
library(tidyverse)
library(smmr)
```

9.1 Measuring body fat

Athletes are concerned with measuring their body fat percentage. Two different methods are available: one using ultrasound, and the other using X-ray technology. We are interested in whether there is a difference in the mean body fat percentage as measured by these two methods, and if so, how big that difference is. Data on 16 athletes are at link.

(a) Explain briefly why a matched pairs analysis is more suitable for these data than a two-independent-samples analysis (using a two-sample t-test). You might find that looking at the data (clicking on the link) helps you figure this out.

(b) Read in the data and check that you have a sensible number of rows and columns.

(c) Carry out a suitable test to determine whether the means are the same or different. (At this point, obtain the R output including a P-value.)

(d) What do you conclude from the test?

(e) Obtain a 95% confidence interval for the population mean difference. How is the interval consistent with your test?

(f) Calculate the differences, and make a normal quantile plot of them. Is there any evidence that normality of differences fails? Explain briefly.

9.2 Throwing baseballs and softballs

Can students throw a baseball farther than a softball? A statistics class, containing 24 students, went out to a football field to try to answer this question. Each student warmed up and then threw each type of ball as far as they could. The order of ball types was randomized: some students threw the baseball first, and some threw the softball first. (A softball is bigger than a baseball, so we might expect that a softball would be harder to throw a long way than a

baseball.) The data are in http://ritsokiguess.site/datafiles/throw.txt in three columns: the first is a number identifying the student, the second is the distance thrown with the baseball (in yards) and the third is the distance thrown with the softball (also in yards).

(a) Read the data into R. You'll need to supply some names to the columns.

(b) Calculate a column of differences, baseball minus softball, *in* the data frame.

(c) Carry out a sign test in R, testing the null hypothesis that the median difference is zero, against the alternative that it is greater than zero. Obtain a P-value. Your option whether you use `smmr` or not.

9.3 Throwing baseballs and softballs, again

Previously, you carried out a sign test to determine whether students could throw a baseball farther than a softball. This time, we will calculate a confidence interval for the median difference baseball minus softball, using the results of sign tests.

(a) Read the data into R from link, giving appropriate names to the columns, and add a column of differences.

(b) Use `smmr` to find a 95% confidence interval for the median difference.

(c) What function in `smmr` will run a two-sided sign test and return only the P-value? Check that it works by testing whether the median difference for your data is zero or different from zero.

(d) Based on your P-value, do you think 0 is inside the confidence interval or not? Explain briefly.

(e) Obtain a 95% confidence interval for the population median difference, baseball minus softball, using a trial-and-error procedure that determines whether a number of possible medians are inside or outside the CI.

9.4 Changes in salary

A company is growing and would like to attract more employees. The company would like to advertise that salaries there are increasing. To do this, the company randomly samples 20 employees that have been working there since January 2016, and for each of these employees, records their salary in January 2016 and January 2017. The data, with salaries in thousands of dollars, are in link.

(a) Read the data into R and demonstrate that you have two salaries for each of 20 employees.

(b) To compare the salaries, explain briefly why a matched-pairs test would be better than a two-sample test.

(c) Make a suitable graph to assess the assumptions for a matched-pairs t-test. What does your graph tell you?

(d) Carry out a suitable matched-pairs t-test on these data. (If you thought in the previous part that this was the wrong thing to do, do it anyway for the purposes of this assignment.) What do you conclude?

(e) The company would like to estimate
how much salaries are increasing, on average. Obtain some output that will enable the company to assess this, and tell the CEO which piece of the output they should look at.

9.5 Body fat revisited

Athletes are concerned with measuring their body fat percentage. Two different methods are available: one using ultrasound, and the other using X-ray technology. We are interested in whether there is a difference in the mean body fat percentage as measured by these two methods, and if so, how big that difference is. Data on 16 athletes are at link.

We saw this data set before.

(a) Read in the data again.

(b) Calculate the differences, and make a normal quantile plot of them. Is there any evidence that normality of differences fails? Explain briefly.

(c) Previously, we did a matched-pairs t-test for these data. In the light of your normal quantile plot, do you think that was a good idea? Explain briefly.

(d) Use the sign test appropriately to compare the two methods for measuring body fat. (Use `smmr` if you wish.) What do you conclude, as ever in the context of the data?

9.6 The dentist and blood pressure

Going to the dentist is scary for a lot of people. One way in which this might show up is that people might have higher blood pressure on average before their dentist's appointment than an hour after the appointment is done. Ten randomly-chosen individuals have their (systolic) blood pressure measured while they are in a dentist's waiting room, and then again one hour after their appointment is finished.

The data are in http://ritsokiguess.site/datafiles/blood_pressure1.csv.

(a) Read in and display the data.

(b) What kind of experimental design is this? How do you know? Explain briefly.

(c) Run a suitable t-test on these data. What do you conclude, in the context of the data?

(d) Run a suitable sign test on these data. What do you conclude, in the context of the data?

(e) Draw a suitable normal quantile plot of these data, one that will enable you to decide between the tests you ran in the previous two parts.

(f) Discuss briefly which of your two tests is the more appropriate one to run.

9.7 French teachers

Twenty high-school French teachers attended a summer institute to improve their French skills. At the beginning of their session, each teacher took a listening test (to test their understanding of spoken French). After 4 weeks of immersion in French, each teacher took a similar listening test again. (The actual French spoken in the two tests was different, so simply taking the first test should not improve the score in the second one; the tests were otherwise similar.) The maximum score on each test was 36, and a higher score is better. The data are here. (Right-click on the link, select "copy link address", and then paste that URL into R Studio.) The data values are separated by *tabs*.

The data file has three columns:

- an identification for each teacher
- the teacher's score in the first test
- the teacher's score in the second test

(a) Read in and display (some of) the data.

(b) Explain briefly why this is a matched-pairs study.

(c) Run a suitable matched-pairs t-test to see whether the teachers' scores have on average *improved* over the four weeks.

(d) What do you conclude from your test, in the context of the data?

(e) How much is the teachers' listening skill improving, on average? Give a suitable interval to support your answer.

(f) Make a suitable plot to assess any assumptions for this test.

(g) Do you trust the result of your matched-pairs t-test? Explain briefly.

(h) Run a suitable sign test, and obtain a suitable (95%) confidence interval. Comment briefly on your results.

(i) Comment briefly on the comparison between your inferences for the mean and the median.

My solutions follow:

9.8 Measuring body fat

Athletes are concerned with measuring their body fat percentage. Two different methods are available: one using ultrasound, and the other using X-ray technology. We are interested in whether there is a difference in the mean body fat percentage as measured by these two methods, and if so, how big that difference is. Data on 16 athletes are at link.

(a) Explain briefly why a matched pairs analysis is more suitable for these data than a two-independent-samples analysis (using a two-sample t-test). You might find that looking at the data (clicking on the link) helps you figure this out.

Solution

The data file looks like this:

```
athlete xray ultrasound
1 5.00 4.75
2 7 3.75
3 9.25 9
4 12 11.75
5 17.25 17
6 29.5 27.5
7 5.5 6.5
8 6 6.75
9 8 8.75
10 8.5 9.5
11 9.25 9.5
12 11 12
13 12 12.25
14 14 15.5
15 17 18
16 18 18.25
```

The data are two measurements for each of the 16 athletes: that is, each athlete had their body fat percentage measured using *both* of the two methods. Extra: a two-sample t approach would be reasonable if one set of 16 athletes had been measured by X-ray and *another different* set of 16 athletes had been measured by ultrasound. (That is, if there had been 32 athletes in total, with each one randomly assigned to *one* of the measurement methods.) But that's not what happened. It is easy to measure one athlete's body fat percentage using both of the two methods, so a matched pairs design is easy to implement (as well as being better). If you use two independent samples (each athlete doing only one measurement method), you introduce an extra source of variability: athletes differ one from another in body fat, as well as differing possibly by measurement method. If you use a matched-pairs design, you remove the athlete-to-athlete differences, leaving only the differences due to measurement method.

■

(b) Read in the data and check that you have a sensible number of rows and columns.

Solution

This kind of thing. Since you looked at the data (didn't you?), you'll know that the values are separated by single spaces:

```
myurl <- "http://ritsokiguess.site/datafiles/bodyfat.txt"
bodyfat <- read_delim(myurl, " ")
```

```
Rows: 16 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (3): athlete, xray, ultrasound

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bodyfat
```

```
# A tibble: 16 x 3
   athlete  xray ultrasound
     <dbl> <dbl>      <dbl>
 1       1  5          4.75
 2       2  7          3.75
 3       3  9.25       9
 4       4 12         11.8
 5       5 17.2       17
```

```
 6        6 29.5        27.5
 7        7  5.5         6.5
 8        8  6           6.75
 9        9  8           8.75
10       10  8.5         9.5
11       11  9.25        9.5
12       12 11          12
13       13 12          12.2
14       14 14          15.5
15       15 17          18
16       16 18          18.2
```

16 rows (athletes) and 3 columns, one for each measurement method and one labelling the athletes. All good.

Since 16 is not too much bigger than 10, I got the whole data frame here. (At least, I think that's the reason I got more than 10 rows.) In an R Notebook, you'll see the first ten rows as normal, with a button to click to see the other six.

■

(c) Carry out a suitable test to determine whether the means are the same or different. (At this point, obtain the R output including a P-value.)

Solution

Feed the two columns into `t.test` along with `paired=T`. This is a two-sided test, so we don't have to take any special steps for that. Note that we're back to the "old-fashioned" version of `t.test` that *does not* allow `data=`, so we have to go the `with` way:

```
with(bodyfat, t.test(xray, ultrasound, paired = TRUE))
```

```
    Paired t-test

data:  xray and ultrasound
t = -0.30801, df = 15, p-value = 0.7623
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.7425068  0.5550068
sample estimates:
mean difference
       -0.09375
```

■

(d) What do you conclude from the test?

Solution

The P-value of 0.7623 is not at all small, so there is no way we can reject the null hypothesis.[1] There is no evidence of a difference in means; we can act as if the two methods produce the same mean body fat percentage. That is to say, on this evidence we can use either method, whichever one is cheaper or more convenient.

■

(e) Obtain a 95% confidence interval for the population mean difference. How is the interval consistent with your test?

Solution

You don't even need to do any more coding: the test was two-sided, so just pick the confidence interval off the output above: -0.74 to 0.56. The interval includes both positive and negative values (or, 0 is inside the interval), so the difference could go either way. This is entirely consistent with not being able to reject the null.

■

(f) Calculate the differences, and make a normal quantile plot of them. Is there any evidence that normality of differences fails? Explain briefly.

Solution

The smoothest[2] way to do this is to use a pipeline: use a `mutate` to create the column of differences, and then pipe that into `ggplot`, omitting the data frame that would normally go first (the input data frame here is the new one with the differences in it, which doesn't have a name). I'll make a normal quantile plot in a moment, but if you haven't seen that yet, the plot to make is a histogram:

```
bodyfat %>%
  mutate(diff = xray - ultrasound) %>%
  ggplot(aes(x = diff)) + geom_histogram(bins = 6)
```

[1] My hat stays on my head.

[2] I learned yesterday that the Welsh word for "ironing" is *smwddio*, which seems weird until you say it out loud: it sounds like "smoothio".

I don't know whether you'd call that "approximately normal" or not. We are in kind of a double-bind with this one: the sample size is small, so normality matters, but with a small sample, the data might not look very normal. It's kind of skewed right, but most of the evidence for the skewness is contained in those two observations with difference 2 and above, which is pretty flimsy evidence for anything. (In the normal quantile plot below, the suggestion is that those two observations really are a bit too large. It's easier to tell there.)

Below, I'm repeating the calculation of the differences, which is inefficient. If I'm going to draw two graphs of the differences, the right way is to calculate the differences *and save the data frame*, then use that new data frame twice. But you're probably only going to draw either the histogram or the normal quantile plot, not both, so you can use the appropriate one of my two bits of code. The normal quantile plot:

```
bodyfat %>%
  mutate(diff = xray - ultrasound) %>%
  ggplot(aes(sample = diff)) + stat_qq() + stat_qq_line()
```

This is showing a little evidence of skewness or outliers (depending on your point of view: either is good). The lowest and highest values are both too high, and the pattern of points on the plot is kind of curved (which would be evidence of skewness). Or you could say that the two highest values are too high, with the other values being more or less in line (that would be evidence of outliers at the upper end). I like outliers better than skewness, since those bottom-end points are not far off the line. I would also accept "no substantial problems", if you can make the case that those two highest points are not too far off the line. With only 16 observations as we have here, even truly normal data would stray off the line a bit.

As ever, your explanation is more important than your conclusion. Can you justify what you think?

If you took your differences the other way around, as `ultrasound` minus `xray`, your plot will also be the other way around, with the "outliers" at the bottom. That's good too.

Where this is going (which I didn't ask you) is whether or not we trust the result of the matched pairs test. I would say that the test is so far from being significant, and the failure of normality is not gross, that it is hard to imagine *any* alternative test coming up with a significant result. So I would be happy to trust this paired t-test.

■

9.9 Throwing baseballs and softballs

Can students throw a baseball farther than a softball? A statistics class, containing 24 students, went out to a football field to try to answer this question. Each student warmed up and then threw each type of ball as far as they could. The order of ball types was randomized: some students threw the baseball first, and some threw the softball first. (A softball is bigger than a baseball, so we might expect that a softball would be harder to throw a long way than a baseball.) The data are in http://ritsokiguess.site/datafiles/throw.txt in three columns: the first is a number identifying the student, the second is the distance thrown with the baseball (in yards) and the third is the distance thrown with the softball (also in yards).

(a) Read the data into R. You'll need to supply some names to the columns.

Solution

This kind of thing:

```
myurl="http://ritsokiguess.site/datafiles/throw.txt"
throws=read_delim(myurl," ",col_names=c("student","baseball","softball"))
```

```
Rows: 24 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (3): student, baseball, softball

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
throws
```

```
# A tibble: 24 x 3
   student baseball softball
     <dbl>    <dbl>    <dbl>
 1       1       65       57
 2       2       90       58
 3       3       75       66
 4       4       73       61
 5       5       79       65
 6       6       68       56
 7       7       58       53
 8       8       41       41
```

```
 9        9        56        44
10       10        70        65
# i 14 more rows
```

This is one of those times where we have to tell R what names to give the columns. Or you can put `col_names=F` and leave the columns called `X1, X2, X3` or whatever they end up as.

■

(b) Calculate a column of differences, baseball minus softball, *in* the data frame.

Solution

Add it to the data frame using `mutate`. Use the right-arrow assignment to create what I called `throws2` below, or put something like `throws2 <-` on the beginning of the line. Your choice.

```
throws %>% mutate(diff=baseball-softball) ->
  throws2
```

■

(c) Carry out a sign test in R, testing the null hypothesis that the median difference is zero, against the alternative that it is greater than zero. Obtain a P-value. Your option whether you use `smmr` or not.

Solution

I think using `smmr` is way easier, so I'll do that first. There is even a shortcut in that the null median defaults to zero, which is exactly what we want here:

```
library(smmr)
sign_test(throws2,diff)
```

```
$above_below
below above
    2    21

$p_values
  alternative      p_value
1       lower 9.999971e-01
2       upper 3.302097e-05
3   two-sided 6.604195e-05
```

We want, this time, the upper-tailed one-sided test, since we want to prove that students can throw a baseball a *longer* distance than a softball. Thus the P-value we want is 0.000033.

To build it yourself, you know the steps by now. First step is to count how many differences are greater and less than zero:

```
table(throws2$diff>0)
```

```
FALSE  TRUE
    3    21
```

or

```
table(throws2$diff<0)
```

```
FALSE  TRUE
   22     2
```

or, since we have things in a data frame,

```
throws2 %>% count(diff>0)
```

```
# A tibble: 2 x 2
  `diff > 0`     n
  <lgl>      <int>
1 FALSE          3
2 TRUE          21
```

or count those less than zero. I'd take any of those.

Note that these are *not all the same*. One of the differences is in fact exactly zero. The technically right thing to do with the zero difference is to throw it away (leaving 23 differences with 2 negative and 21 positive). I would take that, or 2 or 3 negative differences out of 24 (depending on whether you count "greater than zero" or "less than zero"). We hope that this won't make a material difference to the P-value; it'll make some difference, but won't (we hope) change the conclusion about whether to reject.

Second step is to get a P-value for whichever one of those you got, from the appropriate binomial distribution.

The P-value is the probability of getting 21 (or 22) positive differences out of 24 (or 23) or more, since this is the end of the distribution we should be at if the alternative hypothesis is correct. Thus any of these will get you a defensible P-value:

```
sum(dbinom(21:23,23,0.5))
```

```
[1] 3.302097e-05
```

```
sum(dbinom(22:24,24,0.5))
```

```
[1] 1.7941e-05
```

```
sum(dbinom(21:24,24,0.5))
```

```
[1] 0.0001385808
```

```
sum(dbinom(0:2,23,0.5))
```

```
[1] 3.302097e-05
```

```
sum(dbinom(0:2,24,0.5))
```

```
[1] 1.7941e-05
```

```
sum(dbinom(0:3,24,0.5))
```

```
[1] 0.0001385808
```

The first and fourth of those are the same as `smmr` (throwing away the exactly-median value).

As we hoped, there is no *material* difference here: there is no doubt with any of these possibilities that we will reject a median difference of zero in favour of a median difference greater than zero.

■

9.10 Throwing baseballs and softballs, again

Previously, you carried out a sign test to determine whether students could throw a baseball farther than a softball. This time, we will calculate a confidence interval for the median difference baseball minus softball, using the results of sign tests.

(a) Read the data into R from link, giving appropriate names to the columns, and add a column of differences.

Solution

I did it this way, combining the reading of the data with the calculation of the differences in *one* pipe:

```
myurl <- "http://ritsokiguess.site/datafiles/throw.txt"
throws <- read_delim(myurl, " ", col_names = c("student", "baseball", "softball")) %>%
  mutate(diff = baseball - softball)
```

```
Rows: 24 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (3): student, baseball, softball

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
throws
```

```
# A tibble: 24 x 4
   student baseball softball  diff
     <dbl>    <dbl>    <dbl> <dbl>
 1       1       65       57     8
 2       2       90       58    32
 3       3       75       66     9
 4       4       73       61    12
 5       5       79       65    14
 6       6       68       56    12
 7       7       58       53     5
 8       8       41       41     0
 9       9       56       44    12
10      10       70       65     5
# i 14 more rows
```

■

(b) Use `smmr` to find a 95% confidence interval for the median difference.

Solution

`ci_median`, with 95% being the default confidence level:

```
ci_median(throws, diff)
```

```
[1] 2.002930 8.999023
```

2 to 9. The ends of a CI for the median will be data values, which are all whole numbers, so round off that 8.999.

■

(c) What function in `smmr` will run a two-sided sign test and return only the P-value? Check that it works by testing whether the median difference for your data is zero or different from zero.

Solution

The rest of the way, we are trying to reproduce that confidence interval by finding it ourselves. The function is called `pval_sign`. If you haven't run into it before, in R Studio click on Packages, find `smmr`, and click on its name. This will bring up package help, which includes a list of all the functions in the package, along with a brief description of what each one does. (Clicking on a function name brings up the help for that function.) Let's check that it works properly by repeating the previous `sign_test` and verifying that `pval_sign` gives the same thing:

```
sign_test(throws, diff, 0)
```

```
$above_below
below above
    2    21

$p_values
  alternative      p_value
1       lower 9.999971e-01
2       upper 3.302097e-05
3   two-sided 6.604195e-05
```

```r
pval_sign(0, throws, diff)
```

```
[1] 6.604195e-05
```

The P-values are the same (for the two-sided test) and both small, so the median difference is not zero.

■

(d) Based on your P-value, do you think 0 is inside the confidence interval or not? Explain briefly.

Solution

Absolutely not. The median difference is definitely not zero, so zero cannot be in the confidence interval. Our suspicion, from the one-sided test from earlier, is that the differences were mostly positive (people could throw a baseball farther than a softball, in most cases). So the confidence interval ought to contain only positive values. I ask this because it drives what happens below.

■

(e) Obtain a 95% confidence interval for the population median difference, baseball minus softball, using a trial-and-error procedure that determines whether a number of possible medians are inside or outside the CI.

Solution

I've given you a fair bit of freedom to tackle this as you wish. Anything that makes sense is good: whatever mixture of mindlessness, guesswork and cleverness that you want to employ. The most mindless way to try some values one at a time and see what you get, eg.:

```r
pval_sign(1, throws, diff)
```

```
[1] 0.001489639
```

```r
pval_sign(5, throws, diff)
```

```
[1] 1.168188
```

So median 1 is outside and median 5 is inside the 95% interval. Keep trying values until you've figured out where the lower and upper ends of the interval are: where the P-values cross from below 0.05 to above, or vice versa.

Something more intelligent is to make a long list of potential medians, and get the P-value for each of them, eg.:

```
d <- tibble(my.med = seq(0, 20, 2))
d %>% rowwise() %>%
  mutate(pvals = pval_sign(my.med, throws, diff))
```

```
# A tibble: 11 x 2
# Rowwise:
   my.med      pvals
    <dbl>      <dbl>
 1      0 0.0000660
 2      2 0.0525
 3      4 0.839
 4      6 0.678
 5      8 0.210
 6     10 0.0227
 7     12 0.00149
 8     14 0.0000660
 9     16 0.0000359
10     18 0.00000572
11     20 0.00000298
```

2 is just inside the interval, 8 is also inside, and 10 is outside. Some closer investigation:

```
d <- tibble(my.med = seq(0, 2, 0.5))
d %>% rowwise() %>%
  mutate(pvals = pval_sign(my.med, throws, diff))
```

```
# A tibble: 5 x 2
# Rowwise:
  my.med     pvals
   <dbl>     <dbl>
1    0   0.0000660
2    0.5 0.000277
3    1   0.00149
4    1.5 0.0227
5    2   0.0525
```

The bottom end of the interval actually is 2, since 2 is inside and 1.5 is outside.

```
d <- tibble(my.med = seq(8, 10, 0.5))
d %>% rowwise() %>%
  mutate(pvals = pval_sign(my.med, throws, diff))
```

```
# A tibble: 5 x 2
# Rowwise:
  my.med  pvals
   <dbl>  <dbl>
1    8   0.210
2    8.5 0.152
3    9   0.0525
4    9.5 0.0227
5   10   0.0227
```

The top end is 9, 9 being inside and 9.5 outside.

Since the data values are all whole numbers, I think this is accurate enough. The most sophisticated way is the "bisection" idea we saw before. We already have a kickoff for this, since we found, mindlessly, that 1 is outside the interval on the low end and 5 is inside, so the lower limit has to be between 1 and 5. Let's try halfway between, ie. 3:

```
pval_sign(3, throws, diff)
```

```
[1] 0.3833103
```

Inside, so lower limit is between 1 and 3. This can be automated, thus:

```
lo <- 1
hi <- 3
while (abs(hi - lo) > 0.1) {
  try <- (lo + hi) / 2
  ptry <- pval_sign(try, throws, diff)
  if (ptry > 0.05) {
    hi <- try
  } else {
    lo <- try
  }
}
c(lo, hi)
```

```
[1] 1.9375 2.0000
```

The difficult bit is to decide whether the value `try` becomes the new `lo` or the new `hi`. If the P-value for the median of `try` is greater than 0.05, `try` is inside the interval, and it becomes the new `hi`; otherwise it's outside and becomes the new `lo`. Whatever the values are, `lo` is always outside the interval and `hi` is always inside, and they move closer and closer to each other.

At the other end of the interval, `lo` is inside and `hi` is outside, so there is a little switching around within the loop. For starting values, you can be fairly mindless: for example, we know that 5 is inside and something big like 20 must be outside:

```
lo <- 5
hi <- 20
while (abs(hi - lo) > 0.1) {
  try <- (lo + hi) / 2
  ptry <- pval_sign(try, throws, diff)
  if (ptry > 0.05) {
    lo <- try
  } else {
    hi <- try
  }
}
c(lo, hi)
```

```
[1] 8.984375 9.042969
```

The interval goes from 2 to (as calculated here) about 9. (This is apparently the same as `ci_median` in `smmr` got.) `ci_median` uses the bisection method with a smaller "tolerance" than we did, so its answer is more accurate. It looks as if the interval goes from 2 to 9: that is, students can throw a baseball on average between 2 and 9 feet further than they can throw a softball.

■

9.11 Changes in salary

A company is growing and would like to attract more employees. The company would like to advertise that salaries there are increasing. To do this, the company randomly samples 20 employees that have been working there since January 2016, and for each of these employees, records their salary in January 2016 and January 2017. The data, with salaries in thousands of dollars, are in link.

(a) Read the data into R and demonstrate that you have two salaries for each of 20 employees.

Solution

Looking at the file, we see that the values are separated by exactly one space:

```
my_url <- "http://ritsokiguess.site/datafiles/salaryinc.txt"
salaries <- read_delim(my_url, " ")
```

```
Rows: 20 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): employee
dbl (2): jan2016, jan2017

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
salaries
```

```
# A tibble: 20 x 3
   employee jan2016 jan2017
   <chr>      <dbl>   <dbl>
 1 A           36      39.5
 2 B           41.2    47
 3 C           40      45
 4 D           42.8    49
 5 E           51      57.8
 6 F           50.5    54
 7 G           56      62
 8 I           57.8    69.9
 9 J           62      66.8
10 K           65.5    71
11 L           66      72
12 M           68.9    74
13 N           71      80
14 O           72.3    79
15 P           74.6    80
16 Q           77      83.1
17 R           79.9    82.5
18 S           81      92
```

```
19 T           83.2    85
20 U           90     101
```

There are 20 employees (rows), and two columns of salaries: for each employee in the data set, their salary in January 2016 and in January 2017 (thus, two salaries for each employee).

■

(b) To compare the salaries, explain briefly why a matched-pairs test would be better than a two-sample test.

Solution

A matched-pairs test would be better because we have two observations (salaries) for each subject (employee). A two-sample test would be appropriate if we had two separate sets of employees, one set with their salaries recorded in 2016 and the other with their salaries recorded in 2017. That is not what we have here. You can go after this either way: why a matched-pairs approach is appropriate, or why a two-sample approach is not (or a bit of both).

■

(c) Make a suitable graph to assess the assumptions for a matched-pairs t-test. What does your graph tell you?

Solution

This requires thought first before you do any coding (and this is the reason for this one being four points). What has to be at least approximately normally distributed is the set of *differences*, salary at one time point minus the salary at the other, for each employee. The individual salaries don't have to be normally distributed at all. We don't have the differences here, so we have to calculate them first. The smoothest way is to make a pipeline:

```r
salaries %>%
  mutate(diff = jan2017 - jan2016) %>%
  ggplot(aes(sample = diff)) + stat_qq() + stat_qq_line()
```

A couple of coding notes: (i) you can take the differences 2016 minus 2017 if you like (then they will tend to be negative), (ii) `ggplot` used in a pipeline like this does *not* have a data frame first (the data frame used is the nameless output from the `mutate`, with the differences in it).

Also, there's no problem doing the `mutate`, saving that, and then feeding the saved data frame into `ggplot`. If you find that clearer, go for it.

As for what I see: I think those points get a bit far from the line at the high and low ends: the high values are too high and the low values are too low, which is to say that we have outliers at both ends, or the distribution has long tails (either way of saying it is good).

The important conclusion here is whether these differences are normal *enough* to trust a matched pairs t-test here. We have a sample of size 20, so the central limit theorem will help us some, but you can reasonably say that these tails are too long and that we should not trust a matched-pairs t-test.

I actually wanted you to practice doing a matched-pairs t-test anyway, hence my comment in the next part, but the result is probably not *so* trustworthy.

■

(d) Carry out a suitable matched-pairs t-test on these data. (If you thought in the previous part that this was the wrong thing to do, do it anyway for the purposes of this assignment.) What do you conclude?

Solution

The company is trying to prove that salaries are *increasing* over time, so we need a one-sided alternative. Following through the procedure, even though you may not trust it much:

```
with(salaries, t.test(jan2016, jan2017, alternative = "less", paired = TRUE))
```

```
    Paired t-test

data:  jan2016 and jan2017
t = -10.092, df = 19, p-value = 2.271e-09
alternative hypothesis: true mean difference is less than 0
95 percent confidence interval:
      -Inf -5.125252
sample estimates:
mean difference
         -6.185
```

You could also have the years the other way around, in which case the alternative has to be the other way around as well:

```
with(salaries, t.test(jan2017, jan2016, alternative = "greater", paired = TRUE))
```

```
    Paired t-test

data:  jan2017 and jan2016
t = 10.092, df = 19, p-value = 2.271e-09
alternative hypothesis: true mean difference is greater than 0
95 percent confidence interval:
 5.125252      Inf
sample estimates:
mean difference
          6.185
```

Or, if you saved the data frame with the differences in it, do a one-sample test on those, again making sure that you get the `alternative` right. I didn't save it, so I'm calculating the differences again:

```r
salaries %>%
  mutate(diff = jan2017 - jan2016) %>%
  with(., t.test(diff, mu = 0, alternative = "greater"))
```

```
    One Sample t-test

data:  diff
t = 10.092, df = 19, p-value = 2.271e-09
alternative hypothesis: true mean is greater than 0
95 percent confidence interval:
 5.125252      Inf
sample estimates:
mean of x
    6.185
```

Whichever way you do it, the P-value is the same 2.271×10^{-9}, which is a whole lot less than 0.05, so there is no doubt at all that salaries are increasing.

(Your intuition ought to have expected something like this, because everyone's 2017 salary appears to be greater than their 2016 salary.)

Extra: you might be feeling that we ought to be doing a matched-pairs sign test, which you could do this way:

```r
library(smmr)
salaries %>%
  mutate(diff = jan2017 - jan2016) %>%
  sign_test(diff, 0)
```

```
$above_below
below above
    0    20

$p_values
  alternative      p_value
1       lower 1.000000e+00
2       upper 9.536743e-07
3   two-sided 1.907349e-06
```

and then take the "upper" P-value, which is in the same ballpark as the one from the t-test. So the salaries really are increasing, whether you believe the t-test or not. And note that *every single employee's salary increased*.

(Again, the "missing" data frame in `sign_test` is the nameless one with the differences in it.)

■

(e) The company would like to estimate *how much* salaries are increasing, on average. Obtain some output that will enable the company to assess this, and tell the CEO which piece of the output they should look at.

Solution

A confidence interval. 95% is fine. As before, we have to run `t.test` again because we ran a one-sided test and a confidence interval for us is two-sided:

```
with(salaries, t.test(jan2017, jan2016, paired = TRUE))
```

```
    Paired t-test

data:  jan2017 and jan2016
t = 10.092, df = 19, p-value = 4.542e-09
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 4.902231 7.467769
sample estimates:
mean difference
          6.185
```

Between about $5,000 and about $7,500. This is what to tell the CEO.

■

9.12 Body fat revisited

Athletes are concerned with measuring their body fat percentage. Two different methods are available: one using ultrasound, and the other using X-ray technology. We are interested in whether there is a difference in the mean body fat percentage as measured by these two methods, and if so, how big that difference is. Data on 16 athletes are at link.

We saw this data set before.

(a) Read in the data again.

Solution

This kind of thing. Since you looked at the data (didn't you?), you'll know that the values are separated by single spaces:

```
myurl <- "http://ritsokiguess.site/datafiles/bodyfat.txt"
bodyfat <- read_delim(myurl, " ")
```

```
Rows: 16 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (3): athlete, xray, ultrasound

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bodyfat
```

```
# A tibble: 16 x 3
   athlete  xray ultrasound
     <dbl> <dbl>      <dbl>
 1       1  5          4.75
 2       2  7          3.75
 3       3  9.25       9
 4       4 12         11.8
 5       5 17.2       17
 6       6 29.5       27.5
 7       7  5.5        6.5
 8       8  6          6.75
 9       9  8          8.75
10      10  8.5        9.5
11      11  9.25       9.5
12      12 11         12
13      13 12         12.2
14      14 14         15.5
15      15 17         18
16      16 18         18.2
```

■

(b) Calculate the differences, and make a normal quantile plot of them. Is there any evidence that normality of differences fails? Explain briefly.

Solution

This is a good place to look ahead. We'll need the differences in two places, most likely: first for the normal quantile plot, and second for the matched-pairs sign test. So we should calculate and save them first:

```
bodyfat %>% mutate(diff = xray - ultrasound) -> bodyfat2
```

I seem to be using a 2 on the end to name my dataframe-with-differences, but you can use whatever name you like.

Then, not forgetting to use the data frame that we just made:

```
ggplot(bodyfat2, aes(sample = diff)) + stat_qq() + stat_qq_line()
```

This is showing a little evidence of skewness or outliers (depending on your point of view: either is good). The lowest and highest values are both too high, and the pattern of points on the plot is kind of curved (which would be evidence of skewness). Or you could say that the two highest values are too high, with the other values being more or less in line (that would be evidence of outliers at the upper end). I like outliers better than skewness, since those bottom-end points are not far off the line. I would also accept "no substantial problems", if

you can make the case that those two highest points are not too far off the line. With only 16 observations as we have here, even truly normal data would stray off the line a bit.

As ever, your explanation is more important than your conclusion. Can you justify what you think?

If you took your differences the other way around, as `ultrasound` minus `xray`, your plot will also be the other way around, with the "outliers" at the bottom. That's good too.

■

(c) Previously, we did a matched-pairs t-test for these data. In the light of your normal quantile plot, do you think that was a good idea? Explain briefly.

Solution

We are looking for the differences to be approximately normal, bearing in mind that we have a sample of size 16, which is not that large. Say what you think here; the points, if I were giving any here, would be for the way in which you support it. The comment I made before when we did a matched-pairs t-test was that the P-value was so large and non-significant that it was hard to imagine any other test giving a significant result. Another way of saying that is that I considered these differences to be "normal enough", given the circumstances. You might very well take a different view. You could say that these differences are clearly not normal, and that the sample size of 16 is not large enough to get any substantial help from the Central Limit Theorem. From that point of view, running the t-test is clearly not advisable.

■

(d) Use the sign test appropriately to compare the two methods for measuring body fat. (Use `smmr` if you wish.) What do you conclude, as ever in the context of the data?

Solution

That means using a sign test to test the null hypothesis that the median difference is zero, against the alternative that it is not zero. (I don't see anything here to indicate that we are looking only for positive or only for negative differences, so I think two-sided is right. You need some reason to do a one-sided test, and there isn't one here.)

Remembering again to use the data frame that has the differences in it:

```
sign_test(bodyfat2, diff, 0)
```

```
$above_below
below above
   10     6
```

```
$p_values
  alternative   p_value
1       lower 0.2272491
2       upper 0.8949432
3   two-sided 0.4544983
```

The two-sided P-value is 0.4545, so we are nowhere near rejecting the null hypothesis that the median difference is zero. There is no evidence that the two methods for measuring body fat show any difference on average.

The table of aboves and belows says that there were 6 positive differences and 10 negative ones. This is not far from an even split, so the lack of significance is entirely what we would expect.

Extra: this is the same conclusion that we drew the last time we looked at these data (with a matched-pairs t-test). That supports what I said then, which is that the t-test was so far from being significant, that it could be very wrong without changing the conclusion. That is what seems to have happened.

■

9.13 The dentist and blood pressure

Going to the dentist is scary for a lot of people. One way in which this might show up is that people might have higher blood pressure on average before their dentist's appointment than an hour after the appointment is done. Ten randomly-chosen individuals have their (systolic) blood pressure measured while they are in a dentist's waiting room, and then again one hour after their appointment is finished.

The data are in http://ritsokiguess.site/datafiles/blood_pressure1.csv.

(a) Read in and display the data.

Solution

```r
my_url <- "http://ritsokiguess.site/datafiles/blood_pressure1.csv"
blood_pressure <- read_csv(my_url)
```

```
Rows: 10 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): person
dbl (2): before, after
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
blood_pressure
```

```
# A tibble: 10 x 3
   person before after
   <chr>   <dbl> <dbl>
 1 p1        132   118
 2 p2        135   137
 3 p3        149   140
 4 p4        133   139
 5 p5        119   107
 6 p6        121   116
 7 p7        128   122
 8 p8        132   124
 9 p9        119   115
10 p10       110   103
```

Aside: A blood pressure is usually given as two numbers, like "120 over 80''. The first number, which is the one shown in our data, is called the systolic blood pressure. It is the pressure in the arteries when the heart is pumping. The second is called the diastolic blood pressure, and it is the pressure in the arteries when the heart is resting.

■

(b) What kind of experimental design is this? How do you know? Explain briefly.

Solution

This is a matched pairs design. We know this because we have two measurements on each person, or the same people were measured before and after seeing the dentist. (The thing that it is *not* is one group of people measured before seeing the dentist, and a *different* group of people measured afterwards, so a two-sample test is *not* the right thing.)

■

(c) Run a suitable t-test on these data. What do you conclude, in the context of the data?

Solution

A matched-pairs t-test, then. Remember, we want to see whether blood pressure is *lower* afterwards (that is, before is *greater* than after), so this needs to be one-sided:

```r
with(blood_pressure, t.test(before, after, alternative = "greater", paired = TRUE))
```

```

    Paired t-test

data:  before and after
t = 2.9945, df = 9, p-value = 0.007545
alternative hypothesis: true mean difference is greater than 0
95 percent confidence interval:
 2.210659      Inf
sample estimates:
mean difference
            5.7
```

There are some variations possible here: `before` and `after` could be switched (in which case `alternative` must be reversed also).

Or, you can do a one-sample t on the differences, with the right `alternative` corresponding to the way you took differences. If you are looking ahead, you might realize that working out the differences *now* and adding them to the dataframe will be a good idea:

```r
blood_pressure %>%
mutate(difference = before - after) -> blood_pressure
blood_pressure
```

```
# A tibble: 10 x 4
   person before after difference
   <chr>   <dbl> <dbl>      <dbl>
 1 p1        132   118         14
 2 p2        135   137         -2
 3 p3        149   140          9
 4 p4        133   139         -6
 5 p5        119   107         12
 6 p6        121   116          5
 7 p7        128   122          6
 8 p8        132   124          8
 9 p9        119   115          4
10 p10       110   103          7
```

I took the differences this way around since I was expecting, if anything, the before numbers to be bigger than the after ones. And then:

```r
with(blood_pressure, t.test(difference, mu = 0, alternative = "greater"))
```

```
    One Sample t-test

data:  difference
t = 2.9945, df = 9, p-value = 0.007545
alternative hypothesis: true mean is greater than 0
95 percent confidence interval:
 2.210659      Inf
sample estimates:
mean of x
      5.7
```

If you did the differences the other way around, your `alternative` will need to be the other way around also.

The P-value (either way) is 0.008,[3] so we have evidence that the mean blood pressure before is greater than the mean blood pressure after.

■

(d) Run a suitable sign test on these data. What do you conclude, in the context of the data?

Solution

A sign test on the differences. By this point, you will realize that you will need to have obtained the differences. Get them here if you did not already get them:

```r
sign_test(blood_pressure, difference, 0)
```

```
$above_below
below above
    2     8

$p_values
  alternative   p_value
1       lower 0.9892578
2       upper 0.0546875
3   two-sided 0.1093750
```

[3] Give the P-value, and round it off to about this accuracy so that your reader can see easily (i) how it compares to 0.05, and (ii) about how big it is. More than two decimal places is too many.

This one gives us all three P-values. The way around I found the differences, the one we want is "upper", 0.055. There is not quite evidence that median blood pressure before is higher.

■

(e) Draw a suitable normal quantile plot of these data, one that will enable you to decide between the tests you ran in the previous two parts.

Solution

The *differences* are supposed to be approximately normal if a matched-pairs t-test is the thing:

```
ggplot(blood_pressure, aes(sample=difference)) + stat_qq() + stat_qq_line()
```

■

(f) Discuss briefly which of your two tests is the more appropriate one to run.

Solution

Make a call about whether the differences are normal enough. You have a couple of angles you can take:

- the lowest two values are too low, so we have two outliers at the low end

- the lowest *and* highest values are too extreme, so that we have a long-tailed distribution

Either of these would suggest a non-normal distribution, which I think you have to conclude from this plot.

The best answer also considers the sample size: there are only 10 differences, a small sample size, and so we will not get much help from the Central Limit Theorem (the sample size is likely not enough[4] to overcome those two outliers or the long tails). Thus, we should *not* trust the t-test and should prefer the sign test.

Extra: you might be disappointed to go through this and come to the conclusion that there was not a decrease in blood pressure between before and after.

What has happened, I think, is that we have only a small sample (10 people), and having 8 positive differences and 2 negative ones is not quite unbalanced enough (with such a small sample) to rule out chance: that is to say, a median difference of zero. The t-test accounted for the size of the differences, and if you believed the normality was satisfactory, you could demonstrate a difference between before and after. But if you didn't like the normality, you were out of luck: the only test you have is an apparently not very powerful one.

If you wanted to, you could bootstrap the sampling distribution of the sample mean and see how normal it looks:

```
tibble(sim =  1:10000) %>%
  rowwise() %>%
  mutate(the_sample = list(sample(blood_pressure$difference, replace = TRUE))) %>%
  mutate(the_mean = mean(the_sample)) %>%
  ggplot(aes(sample = the_mean)) + stat_qq() + stat_qq_line()
```

[4]But see the Extra.

(Code note: you can do anything with the result of a simulation, and you can use anything that might need to be normal as input to a normal quantile plot. Now that we have the normal quantile plot as a tool, we can use it wherever it might be helpful.)

This is actually not nearly as bad as I was expecting. Even a sample size of 10 is providing some help. The bootstrapped sampling distribution is somewhat left-skewed, which is not a surprise given the two low outliers. However, it is rather close to normal, suggesting that the t-test is not as bad as we thought.

(I did 10,000 simulations because I was having trouble seeing how non-normal it was. With this many, I can be pretty sure that this distribution is somewhat left-skewed.)

■

9.14 French teachers

Twenty high-school French teachers attended a summer institute to improve their French skills. At the beginning of their session, each teacher took a listening test (to test their understanding of spoken French). After 4 weeks of immersion in French, each teacher took a similar listening test again. (The actual French spoken in the two tests was different, so simply taking the first test should not improve the score in the second one; the tests were otherwise similar.) The maximum score on each test was 36, and a higher score is better. The data are here.

(Right-click on the link, select "copy link address", and then paste that URL into R Studio.) The data values are separated by *tabs*.

The data file has three columns:

- an identification for each teacher
- the teacher's score in the first test
- the teacher's score in the second test

(a) Read in and display (some of) the data.

Solution

Separated by tabs means `read_tsv`:

```
my_url <- "http://ritsokiguess.site/datafiles/frenchtest.txt"
french <- read_tsv(my_url)
```

```
Rows: 20 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: "\t"
dbl (3): id, pre, post

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
french
```

```
# A tibble: 20 x 3
      id   pre  post
   <dbl> <dbl> <dbl>
 1     1    32    34
 2     2    31    31
 3     3    29    35
 4     4    10    16
 5     5    30    33
 6     6    33    36
 7     7    22    24
 8     8    25    28
 9     9    32    26
10    10    20    26
11    11    30    36
```

```
12     12     20     26
13     13     24     27
14     14     24     24
15     15     31     32
16     16     30     31
17     17     15     15
18     18     32     34
19     19     23     26
20     20     23     26
```

As promised. The score on the first test is called `pre` and on the second is called `post`.

■

(b) Explain briefly why this is a matched-pairs study.

Solution

There are two measurements for each teacher, or, the 20 `pre` measurements and the 20 `post` measurements are paired up, namely, the ones that come from the same teacher. Or, if it were two independent samples, our 40 measurements would come from 40 different teachers, but there are only 20 teachers, so the 40 measurements must be paired up.

■

(c) Run a suitable matched-pairs t-test to see whether the teachers' scores have on average *improved* over the four weeks.

Solution

Seeing whether the scores have improved implies a *one*-sided test that `post` is bigger than `pre`. There are three ways you might do that, any of which is good. Remember that if you are running a test with `paired = TRUE`, the alternative is relative to the column that is *input first*, not the first one in alphabetical order or anything like that:

(i):

```r
with(french, t.test(pre, post, paired = TRUE, alternative = "less"))
```

```

    Paired t-test

data:  pre and post
t = -3.8649, df = 19, p-value = 0.0005216
alternative hypothesis: true mean difference is less than 0
```

```
95 percent confidence interval:
      -Inf -1.381502
sample estimates:
mean difference
           -2.5
```

(ii):

```r
with(french, t.test(post, pre, paired = T, alternative = "greater"))
```

```

    Paired t-test

data:  post and pre
t = 3.8649, df = 19, p-value = 0.0005216
alternative hypothesis: true mean difference is greater than 0
95 percent confidence interval:
 1.381502      Inf
sample estimates:
mean difference
            2.5
```

Your choice between these two might be influenced by whether you think `pre` comes first, or whether you think it's easier to decide how `post` compares to `pre`. It's all down to what seems natural to you.

(iii) working out the differences and testing those (but look ahead in the question to see whether you need the differences for anything else: you do):

```r
french %>% mutate(gain = post - pre) -> french1
with(french1, t.test(gain, mu=0, alternative = "greater"))
```

```

    One Sample t-test

data:  gain
t = 3.8649, df = 19, p-value = 0.0005216
alternative hypothesis: true mean is greater than 0
95 percent confidence interval:
 1.381502      Inf
```

```
sample estimates:
mean of x
      2.5
```

This last is an ordinary one-sample test, which saves you having to learn anything new, but requires you to calculate the differences first. You will need the differences for a plot anyway, so this may not be as much extra work as it appears. The right thing to do here is to *save* the data frame with the differences in it, so that you don't need to calculate them again later.

A fourth alternative is to calculate the differences as `pre` minus `post`, and then switch the `alternative` around (since if going to the French institute helps, the differences this way will be mostly *negative*):

```
french %>% mutate(gain = pre - post) -> french2
with(french2, t.test(gain, mu=0, alternative = "less"))
```

```
    One Sample t-test

data:  gain
t = -3.8649, df = 19, p-value = 0.0005216
alternative hypothesis: true mean is less than 0
95 percent confidence interval:
      -Inf -1.381502
sample estimates:
mean of x
     -2.5
```

■

(d) What do you conclude from your test, in the context of the data?

Solution

The P-value of 0.0005 is much less than 0.05, so we reject the null hypothesis that the mean scores before and after are the same, in favour of the alternative that the mean score afterwards is higher. That is to say, the four-week program is helping the teachers improve their understanding of spoken French.

■

(e) How much is the teachers' listening skill improving, on average? Give a suitable interval to support your answer.

Solution

A 95% (or other level) confidence interval for the mean difference. A one-sided test doesn't give that, so you need to do the test again without the `alternative` (to make it two-sided), via any of the methods above, such as:

```
with(french, t.test(post, pre, paired = TRUE))
```

```

    Paired t-test

data:  post and pre
t = 3.8649, df = 19, p-value = 0.001043
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 1.146117 3.853883
sample estimates:
mean difference
            2.5
```

This says that, with 95% confidence, the mean test score afterwards is between about 1.1 and 3.9 points higher than before. So that's how much listening skill is improving on average. Give the suitably rounded interval; the test scores are whole numbers, and there are 20 differences making up the mean, so one decimal is the most you should give.

If you did it the first way:

```
with(french, t.test(pre, post, paired = TRUE))
```

```

    Paired t-test

data:  pre and post
t = -3.8649, df = 19, p-value = 0.001043
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -3.853883 -1.146117
sample estimates:
mean difference
           -2.5
```

you have given yourself a bit of work to do, because this is before minus after, so you have to strip off the minus signs and switch the numbers around. Giving the answer with the minus signs is wrong, because I didn't ask about before minus after. Disentangle it, though, and you're good.

■

(f) Make a suitable plot to assess any assumptions for this test.

Solution

The key assumption here is that the *differences* are approximately normally distributed.

First calculate *and save* the differences (since you will need them later for a sign test; otherwise you would have to find them again). If you found the differences to make your t-test, use the ones you saved there.

```
french %>% mutate(gain = post - pre) -> french1
```

Assess that with a histogram (with suitable number of bins):

```
ggplot(french1, aes(x=gain)) + geom_histogram(bins=6)
```

or, better, a normal quantile plot (since the normality is our immediate concern):

```r
ggplot(french1, aes(sample=gain)) + stat_qq() + stat_qq_line()
```

(note that the horizontal lines of points are because the test scores were whole numbers, therefore the differences between them are whole numbers also, and some of the teachers had the same difference in scores as others.)

■

(g) Do you trust the result of your matched-pairs t-test? Explain briefly.

Solution

There are about three considerations here:

- the plot shows an outlier at the low end, but no other real problems.
- the sample size is 20, so we should get some help from the Central Limit Theorem.
- the P-value was really small.

I expect you to mention the first two of those. Make a call about whether you think that outlier is too much of a problem, given the sample size. You could, I think, go either way with this one.

The third of my points says that even if the distribution of differences is not normal enough, and so the P-value is off by a bit, it would take a lot to change it enough to stop it being significant. So I don't think we need to worry, for myself.

Extra:

We can assess the t-test by obtaining a bootstrap distribution of the sample mean, by sampling from the differences with replacement:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(french1$gain, replace = T))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```

The bootstrapped sampling distribution of the sample mean difference is about as normal as you could reasonably wish for, so there was no need to worry. Only a very few of the most extreme samples were at all off the line.

A histogram would be almost as good, but now that you know about the normal quantile plot, the time to use it is when you are *specifically* interested in normality, as you are here. (If you were interested in shape generally, then a histogram or, if appropriate, a boxplot, would also work.)

The code: the first line takes 1000 bootstrap samples, and the second finds the mean of each one. Instead of saving the sample means, since I was only going to be using them once, I made them into a dataframe, and then made a normal quantile plot of them. The `enframe` creates a dataframe with a column called `value` with the means in it, which I use in the plot.

■

(h) Run a suitable sign test, and obtain a suitable (95%) confidence interval. Comment briefly on your results.

Solution

This works with the differences, that you calculated for the plot, so use the data frame that you saved them in:

```
sign_test(french1, gain, 0)
```

```
$above_below
below above
    1    16

$p_values
  alternative      p_value
1       lower 0.9999923706
2       upper 0.0001373291
3   two-sided 0.0002746582
```

```
ci_median(french1, gain)
```

```
[1] 1.007812 3.000000
```

The P-value is 0.00014, again very small, saying that the median difference is greater than zero, that is, that the test scores after are greater than the test scores before on average. The confidence interval is from 1 to 3 points, indicating that this is how much test scores are increasing on average.

A technique thing: the first time you are going through this, you probably got to this point and realized that you were calculating the differences for the second (or third) time. This is the place to stop and think that you don't really need to do that, and to go back to the plot you did and *save* the differences after you have calculated them. Then you edit the code here to use the differences you got before and saved. It doesn't matter whether you see this the first time you do it or not, but it does matter that you see it before you hand it in. It's like editing an essay; you need to go back through work that you will be handing in and make sure you did it the best way you could.

■

(i) Comment briefly on the comparison between your inferences for the mean and the median.

Solution

The upper-tail P-value is 0.0001, in the same ballpark as the t-test (0.0005). The 95% confidence interval for the median difference is from 1 to 3,[5] again much like the t-interval (1.1 to 3.9).

This suggests that it doesn't matter much which test we do, and therefore that the t-test ought to be better because it uses the data better.[6] This is more evidence that the outlier didn't have that big of an effect.

Extra: choosing a test on the basis of its P-value is *wrong*, because as soon as you introduce a choice on that basis, your P-value looks lower than it should; a P-value is based on you doing one test and only that one test. It is reasonable to note, as I did, that the two P-values are about the same and then choose between the tests on *some other basis*, such as that the t-test uses the data better.[7]

■

[5] I think I mentioned elsewhere that the P-value of the sign test, as it depends on the null median for a fixed data set, only changes at a data point. Therefore, the ends of a CI for the median must be data points.

[6] It uses the actual data values, not just whether each one is positive or negative.

[7] If the P-values had come out very different, that would be telling you that it matters which one you use, and you would need to go back and look at your plot to decide. Often, this happens when there is something wrong with the t-test, but not necessarily.

10 Normal quantile plots

```
library(tidyverse)
```

10.1 Lengths of heliconia flowers

The tropical flower *Heliconia* is fertilized by hummingbirds, a different species for each variety of Heliconia. Over time, the lengths of the flowers and the form of the hummingbirds' beaks have evolved to match each other. The length of the Heliconia flower is therefore an important measurement. Does it have a normal distribution for each variety?

The data set at http://ritsokiguess.site/datafiles/heliconia.csv contains the lengths (in millimetres) of samples of flowers from each of three varieties of Heliconia: *bihai*, *caribaea* red, and *caribaea* yellow.

(a) Read the data into R. There are different numbers of length measurements for each variety. How does this show up in the data frame? (Look at all the rows, not just the first ten.)

(b) Make a normal quantile plot for the variety *bihai*.

(c) Make a normal quantile plot for the variety *Caribaea* red (note that the variable name in the data frame has an underscore in it).

(d) Make a normal quantile plot for the variety *Caribaea* yellow (this also has an underscore in it).

(e) Which of the three varieties is closest to having a normal distribution? Explain (very) briefly.

(f) For each of the two other varieties, apart from the one you mentioned in the last part, describe briefly how their distributions fail to be normal.

10.2 Ferritin and normality

In the lecture notes, we looked at some data on different athletes from the Australian Institute of Sport. This data set can be found at http://ritsokiguess.site/datafiles/ais.txt. Recall that the values are separated by *tabs*. In this question, we will assess one of the variables in the data set for normality.

(a) Read the data set into R.

(b) One of the variables, `Ferr`, is a measurement of Ferritin for each athlete. Obtain a normal quantile plot of the Ferritin values, for all the athletes together. What do you conclude about the shape of the distribution? Explain briefly.

(c) It is possible that the shape you found in the previous part is because the athletes from all the different sports were mixed together. Use `ggplot` to obtain one normal quantile plot for each sport, collected together on one plot.

(d) Looking at the plots in the previous part, would you say that the Ferritin values for each sport individually have a more normal shape than they do for all the sports together? Explain briefly.

My solutions follow:

10.3 Lengths of heliconia flowers

The tropical flower *Heliconia* is fertilized by hummingbirds, a different species for each variety of Heliconia. Over time, the lengths of the flowers and the form of the hummingbirds' beaks have evolved to match each other. The length of the Heliconia flower is therefore an important measurement. Does it have a normal distribution for each variety?

The data set at http://ritsokiguess.site/datafiles/heliconia.csv contains the lengths (in millimetres) of samples of flowers from each of three varieties of Heliconia: *bihai*, *caribaea* red, and *caribaea* yellow.

(a) Read the data into R. There are different numbers of length measurements for each variety. How does this show up in the data frame? (Look at all the rows, not just the first ten.)

Solution

The usual `read_csv`:

```
my_url <- "http://ritsokiguess.site/datafiles/heliconia.csv"
heliconia <- read_csv(my_url)
```

```
Rows: 23 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (3): bihai, caribaea_red, caribaea_yellow

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

I suggested to look at *all* the rows. Here's why:

```
heliconia
```

```
# A tibble: 23 x 3
   bihai caribaea_red caribaea_yellow
   <dbl>        <dbl>           <dbl>
 1  47.1         41.9            36.8
 2  46.8         42.0            37.0
 3  46.8         41.9            36.5
 4  47.1         43.1            36.1
 5  46.7         41.5            36.0
 6  47.4         41.7            35.4
 7  46.4         39.8            38.1
 8  46.6         40.6            37.1
 9  48.1         39.6            35.2
10  48.3         42.2            36.8
# i 13 more rows
```

The varieties with fewer values have missings (NAs) attached to the end. This is because all the columns in a data frame have to have the same number of values. (The missings won't impact what we do below — we get a warning but not an error, and the plots are the same as they would be without the missings — but you might be aesthetically offended by them, in which case you can read what I do later on.)

■

(b) Make a normal quantile plot for the variety *bihai*.

Solution

There's a certain amount of repetitiveness here (that we work around later):

```
ggplot(heliconia,aes(sample=bihai))+stat_qq()+stat_qq_line()
```

```
Warning: Removed 7 rows containing non-finite values (`stat_qq()`).

Warning: Removed 7 rows containing non-finite values (`stat_qq_line()`).
```

I'm saving the comments until we've seen all three.

■

(c) Make a normal quantile plot for the variety *Caribaea* red (note that the variable name in the data frame has an underscore in it).

Solution

Same idea again:

```
ggplot(heliconia,aes(sample=caribaea_red))+stat_qq()+stat_qq_line()
```

■

(d) Make a normal quantile plot for the variety *Caribaea* yellow (this also has an underscore in it).

Solution

And, one more time:

```
ggplot(heliconia,aes(sample=caribaea_yellow))+stat_qq()+stat_qq_line()
```

```
Warning: Removed 8 rows containing non-finite values (`stat_qq()`).

Warning: Removed 8 rows containing non-finite values (`stat_qq_line()`).
```

I did a lot of copying and pasting there.

■

(e) Which of the three varieties is closest to having a normal distribution? Explain (very) briefly.

Solution

Look at the three plots, and see which one stays closest to the line. To my mind, this is clearly the last one, *Caribaea* yellow. So your answer ought to be "*Caribaea* yellow, because the points are closest to the line". This, I would say, is acceptably close to normal, so using a t-test here would be fine. The answer "the last one" is not quite complete, because I asked you which *variety*, so your answer needs to name a variety.

■

(f) For each of the two other varieties, apart from the one you mentioned in the last part, describe briefly how their distributions fail to be normal.

Solution

Let's look at *bihai* first. I see this one as an almost classic curve: the points are above the line, then below, then above again. If you look at the data scale (y-axis), the points are too bunched up to be normal at the bottom, and too spread out at the top: that is, skewed to the *right*. You might also (reasonably) take the view that the points at the bottom are close to the

line (not sure about the very smallest one, though), but the points at the top are farther away, so that what we have here is two outliers at the top. I'm OK with that. It's often difficult to distinguish between skewness and outliers (at the end of the long tail). What you conclude can often depend on how you look. We also had to look at the second plot, *caribaea* red. This is a rather strange one: the points veer away from the line at the ends, but look carefully: it is *not* outliers at both ends, but rather the points are *too bunched up* to be normal at both ends: that is, the distribution has *short tails* compared to the normal. It is something more like a uniform distribution, which has no tails at all, than a normal distribution, which won't have outliers but it *does* have some kind of tails. So, "short tails".

Extra: that's all you needed, but I mentioned above that you might have been offended aesthetically by those missing values that were not really missing. Let's see if we can do this aesthetically. As you might expect, it uses several of the tools from the "tidyverse". First, tidy the data. The three columns of the data frame are all lengths, just lengths of different things, which need to be labelled. This is `pivot_longer` from `tidyr`:

```
heliconia %>%
  pivot_longer(everything(), names_to="variety", values_to="length", values_drop_na = T) -
heliconia.long
```

```
# A tibble: 54 x 2
   variety         length
   <chr>            <dbl>
 1 bihai             47.1
 2 caribaea_red      41.9
 3 caribaea_yellow   36.8
 4 bihai             46.8
 5 caribaea_red      42.0
 6 caribaea_yellow   37.0
 7 bihai             46.8
 8 caribaea_red      41.9
 9 caribaea_yellow   36.5
10 bihai             47.1
# i 44 more rows
```

This is now aesthetic as well as tidy: all those `NA` lines have gone (you can check that there are now $16 + 23 + 15 = 54$ rows of actual data, as there should be). This was accomplished by the last thing in the `pivot_longer`: "in the values (that is, the lengths), drop any missing values."

Now, how to get a normal quantile plot for each variety? This is `facet_wrap` on the end of the `ggplot` again.

```
ggplot(heliconia.long,aes(sample=length))+
stat_qq()+stat_qq_line()+
facet_wrap(~variety,scale="free")
```

These are a bit elongated vertically. The `scale="free"` allows a different vertical scale for each plot (otherwise there would be one vertical scale for all three plots); I decided that was best here since the typical lengths for the three varieties are different. *Caribaea* yellow is more or less straight, *bihai* has outliers (and may also be curved), *caribaea* red has that peculiar S-bend shape.

I didn't really like the vertical elongation. I'd rather have the plots be almost square, which they would be if we put them in three cells of a 2×2 grid. `facet_wrap` has `nrow` and `ncol` which you can use one or both of to make this happen. This creates an array of plots with two columns and as many rows as needed:

```
ggplot(heliconia.long,aes(sample=length))+
stat_qq()+stat_qq_line()+
facet_wrap(~variety,scale="free",ncol=2)
```

I think the squarer plots make it easier to see the shape of these: curved, S-bend, straightish. Almost the same code will get a histogram for each variety, which I'll also make squarish:

```
ggplot(heliconia.long,aes(x=length))+
geom_histogram(bins=5)+facet_wrap(~variety,scale="free",ncol=2)
```

bihai has those two outliers, *caribaea* red has no tails to speak of (or you might say "it's bimodal", which would be another explanation of the pattern on the normal quantile plot[1]) and *caribaea* yellow is shoulder-shruggingly normal (I looked at that and said, "well, I *guess* it's normal".) After you've looked at the normal quantile plots, you see what a crude tool a histogram is for assessing normality.

■

10.4 Ferritin and normality

In the lecture notes, we looked at some data on different athletes from the Australian Institute of Sport. This data set can be found at http://ritsokiguess.site/datafiles/ais.txt. Recall that the values are separated by *tabs*. In this question, we will assess one of the variables in the data set for normality.

(a) Read the data set into R.

Solution

`read_tsv` is the right thing:

[1] If you have studied a thing called kurtosis, the fourth moment about the mean, you'll know that this measures *both* tail length *and* peakedness, so a short-tailed distribution also has a strong peak. Or, maybe, in this case, two strong peaks.

```r
my_url="http://ritsokiguess.site/datafiles/ais.txt"
athletes=read_tsv(my_url)
```

```
Rows: 202 Columns: 13
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr  (2): Sex, Sport
dbl (11): RCC, WCC, Hc, Hg, Ferr, BMI, SSF, %Bfat, LBM, Ht, Wt

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
athletes
```

```
# A tibble: 202 x 13
   Sex    Sport     RCC   WCC    Hc    Hg  Ferr   BMI   SSF `%Bfat`   LBM    Ht
   <chr>  <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
 1 female Netball  4.56  13.3  42.2  13.6    20  19.2  49      11.3  53.1  177.
 2 female Netball  4.15   6    38    12.7    59  21.2 110.     25.3  47.1  173.
 3 female Netball  4.16   7.6  37.5  12.3    22  21.4  89      19.4  53.4  176
 4 female Netball  4.32   6.4  37.7  12.3    30  21.0  98.3    19.6  48.8  170.
 5 female Netball  4.06   5.8  38.7  12.8    78  21.8 122.     23.1  56.0  183
 6 female Netball  4.12   6.1  36.6  11.8    21  21.4  90.4    16.9  56.4  178.
 7 female Netball  4.17   5    37.4  12.7   109  21.5 107.     21.3  53.1  177.
 8 female Netball  3.8    6.6  36.5  12.4   102  24.4 157.     26.6  54.4  174.
 9 female Netball  3.96   5.5  36.3  12.4    71  22.6 101.     17.9  56.0  174.
10 female Netball  4.44   9.7  41.4  14.1    64  22.8 126.     25.0  51.6  174.
# i 192 more rows
# i 1 more variable: Wt <dbl>
```

I listed the data to check that I had it right, but I didn't ask you to. (If you didn't have it right, that will show up soon enough.)

■

(b) One of the variables, `Ferr`, is a measurement of Ferritin for each athlete. Obtain a normal quantile plot of the Ferritin values, for all the athletes together. What do you conclude about the shape of the distribution? Explain briefly.

Solution

As you would expect:

```
ggplot(athletes, aes(sample=Ferr))+
stat_qq()+stat_qq_line()
```

This is almost a classic right skew: the values are too bunched up at the bottom and too spread out at the top. The curved shape should be making you think "skewed" and then you can work out which way it's skewed.

■

(c) It is possible that the shape you found in the previous part is because the athletes from all the different sports were mixed together. Use `ggplot` to obtain one normal quantile plot for each sport, collected together on one plot.

Solution

Your previous plot had all the sports mixed together. To that you add something that will put each sport in its own facet:

```
ggplot(athletes,aes(sample=Ferr))+stat_qq()+stat_qq_line()+
facet_wrap(~Sport)
```

■

(d) Looking at the plots in the previous part, would you say that the Ferritin values for each sport individually have a more normal shape than they do for all the sports together? Explain briefly.

Solution

There are a couple of ways you can go, and as ever I'm looking mostly for consistency of argument. The two major directions you can go are (i) most of these plots are still curved the same way as the previous one, and (ii) they are mostly straighter than they were before. Possible lines of argument include that pretty much all of these plots are right-skewed still, with the same upward-opening curve. Pretty much the only one that doesn't is Gymnastics, for which there are only four observations, so you can't really tell. So, by this argument, Ferritin just *does* have a right-skewed distribution, and breaking things out by sport doesn't make much difference to that. Or, you could go another way and say that the plot of all the data together was *very* curved, and these plots are much less curved, that is to say, much less skewed. Some of them, such as basketball and netball, are almost straight, and they are almost normally distributed. Some of the distributions, such as track sprinting (`TSprnt`), are definitely still right-skewed, but not as seriously so as before. Decide what you think and then discuss how you see it.

■

11 Analysis of variance

```
library(tidyverse)
```

11.1 Movie ratings and lengths

Before a movie is shown in theatres, it receives a "rating" that says what kind of material it contains. link explains the categories, from G (suitable for children) to R (anyone under 17 must be accompanied by parent/guardian). In 2011, two students collected data on the length (in minutes) and the rating category, for 15 movies of each rating category, randomly chosen from all the movies released that year. The data are at link.

(a) Read the data into R, and display (some of) what you read in.

(b) Count how many movies there are of each rating.

(c) Carry out an ANOVA and a Tukey analysis (if warranted).

(d) Make a graph to assess whether this ANOVA is trustworthy. Discuss your graph and its implications briefly.

11.2 Deer and how much they eat

Do adult deer eat different amounts of food at different times of the year? The data in link are the weights of food (in kilograms) consumed by randomly selected adult deer observed at different times of the year (in February, May, August and November). We will assume that these were different deer observed in the different months. (If the same animals had been observed at different times, we would have been in the domain of "repeated measures", which would require a different analysis, beyond the scope of this course.)

(a) Read the data into R, and calculate numbers of observations and the median amounts of food eaten each month.

(b) Make side-by-side boxplots of the amount of food eaten each month. Comment briefly on what you see.

(c) Run a Mood's median test as in lecture (ie. not using `smmr`). What do you conclude, in the context of the data?

(d) Run a Mood's median test using `smmr`, and compare the results with the previous part.

(e) How is it that Mood's median test does not completely answer the question you really want to answer? How might you get an answer to the question you *really* want answered? Explain briefly, and obtain the answer you *really* want, discussing your results briefly.

11.3 Movie ratings again

This question again uses the movie rating data at link.

(a) Read the data into R and obtain the number of movies of each rating and the *median* length of movies of each rating.

(b) Obtain a suitable graph that assesses the assumptions for ANOVA. Why do you think it is not reasonable to run ANOVA here? Explain briefly.

(c) Run a Mood's median test (use `smmr` if you like). What do you conclude, in the context of the data?

11.4 Atomic weight of carbon

The atomic weight of the chemical element carbon is 12. Two methods of measuring the atomic weight of samples of carbon were compared. The results are shown in link. The methods are labelled 1 and 2. The first task is to find out whether the two methods have different "typical" measures (mean or median, as appropriate) of the atomic weight of carbon.

For this question, compose a report in R Markdown. (R Markdown is what you use in an R Notebook, but you can also have a separate R Markdown document from which you can produce HTML, Word etc. output.) See part (a) for how to get this started.

Your report should read like an actual report, not just the answers to some questions that I set you. To help with that, write some text that links the parts of the report together smoothly, so that it reads as a coherent whole. The grader had 3 discretionary marks to award for the overall quality of your writing. The scale for this was:

- 3 points: excellent writing. The report flows smoothly, is easy to read, and contains everything it should (and nothing it shouldn't).
- 2 points: satisfactory writing. Not the easiest to read, but says what it should, and it looks at least somewhat like a report rather than a string of answers to questions.

- 1 point: writing that is hard to read or to understand. If you get this (or 0), you should consider what you need to do to improve when you write your project.
- 0 points: you answered the questions, but you did almost nothing to make it read like a report.

(a) Create a new R Markdown document. To do this, in R Studio, select File, New File, R Markdown. Type the report title and your name in the boxes, and leave the output on the default HTML. Click OK.

(b) Write an introduction that explains the purpose of this study and the data collected in your own words.

(c) Begin an appropriately-titled new section in your report, read the data into R and display the results.

(d) Make an appropriate plot to compare the measurements obtained by the two methods. You might need to do something about the two methods being given as numbers even though they are really only identifiers. (If you do, your report ought to say what you did and why.)

(e) Comment briefly on what you see in your plot.

(f) Carry out the most appropriate t-test. (You might like to begin another new section in your report here.)

(g) Do the most appropriate test you know that does not assume normally-distributed data.

(h) Discuss the results of your tests and what they say about the two methods for measuring the atomic weight of carbon. If it seems appropriate, put the discussion into a section called Conclusions.

11.5 Can caffeine improve your performance on a test?

Does caffeine help students do better on a certain test? To find out, 36 students were randomly allocated to three groups (12 in each group). Each student received a fixed number of cups of coffee while they were studying, but the students didn't know whether they were receiving all full-strength coffee ("high"), all decaf coffee ("low") or a 50-50 mixture of the two ("moderate"). For each subject, their group was recorded as well as their score on the test. The data are in link, as a `.csv` file.

(a) Read in and examine the data. How are the values laid out?

(b) Explain briefly how the data are not "tidy".

(c) Use a suitable tool from the `tidyverse` to create one column of test scores and and one column of group labels. Call your column of group labels `amount`. Is it a `factor`?

(d) Obtain side-by-side boxplots of test scores by amount of caffeine.

(e) Does caffeine amount seem to have an effect? If so, what kind of effect?

(f) Run a suitable analysis of variance to determine whether the mean test score is equal or unequal for the three groups. What do you conclude?

(g) Why is it a good idea to run Tukey's method here?

(h) Run Tukey's method. What do you conclude?

11.6 Reggae music

Reggae is a music genre that originated in Jamaica in the late 1960s. One of the most famous reggae bands was Bob Marley and the Wailers. In a survey, 729 students were asked to rate reggae music on a scale from 1, "don't like it at all" to 6, "like it a lot". We will treat the ratings as quantitative. Each student was also asked to classify their home town as one of "big city", "suburban", "small town", "rural". Does a student's opinion of reggae depend on the kind of home town they come from? The data are in http://ritsokiguess.site/datafiles/reggae.csv.

(a) Read in and display (some of) the data.

(b) How many students are from each different size of town?

(c) Make a suitable graph of the two variables in this data frame.

(d) Discuss briefly why you might prefer to run Mood's median test to compare ratings among home towns.

(e) Suppose that somebody wanted to run Welch ANOVA on these data. What would be a reasonable argument to support that?

(f) Run Mood's median test and display the output.

(g) Explain briefly why running pairwise median tests is a good idea, run them, and display the results.

(h) Summarize, as concisely as possible, how the home towns differ in terms of their students' ratings of reggae music.

11.7 Watching TV and education

The General Social Survey is a large survey of a large number of people. One of the questions on the survey is "how many hours of TV do you watch in a typical day?" Another is "what is your highest level of education attained", on this scale:

- **HSorLess**: completed no more than high h school
- **College**: completed some form of college, either a community college (like Centennial) or a four-year university (like UTSC)
- **Graduate**: completed a graduate degree such as an MSc.

Do people with more education tend to watch more TV? We will be exploring this. The data are in http://ritsokiguess.site/datafiles/gss_tv.csv.

(a) Read in and display (some of) the data.

(b) For each level of education, obtain the number of observations, the mean and the median of the number of hours of TV watched.

(c) What does your answer to the previous part tell you about the shapes of the distributions of the numbers of hours of TV watched? Explain briefly.

(d) Obtain a suitable graph of your data frame.

(e) Does your plot indicate that your guess about the distribution shape was correct? Explain briefly.

(f) Run a suitable test to compare the average number of hours of TV watched for people with each amount of education. ("Average" could be mean or median, whichever you think is appropriate.)

(g) What do you conclude from your test, in the context of the data?

(h) Why might you now want to run some kind of follow-up test? Run the appropriate thing and explain briefly what you conclude from it, in the context of the data.

11.8 Death of poets

Some people believe that poets, especially female poets, die younger than other types of writer. William Butler Yeats[1] wrote:

[1]An Irish, that is to say, Gaelic, poet (see below), but a male one.

> She is the Gaelic[2] muse, for she gives inspiration to those she persecutes. The Gaelic poets die young, for she is restless, and will not let them remain long on earth.

A literature student wanted to investigate this, and so collected a sample of 123 female writers (of three different types), and noted the age at death of each writer.

The data are in http://ritsokiguess.site/datafiles/writers.csv.

(a) Read in and display (some of) the data.

(b) Make a suitable plot of the ages and types of writing.

(c) Obtain a summary table showing, for each type of writing, the number of writers of that type, along with the mean, median and standard deviation of their ages at death.

(d) Run a complete analysis, starting with an ordinary (not Welch) analysis of variance, that ends with a conclusion in the context of the data and an assessment of assumptions.

11.9 Religion and studying

Many students at a certain university were asked about the importance of religion in their lives (categorized as "not", "fairly", or "very" important), and also about the number of hours they spent studying per week. (This was part of a much larger survey.) We want to see whether there is any kind of relationship between these two variables. The data are in here.

(a) Read in and display (some of) the data.

(b) Obtain the number of observations and the mean and standard deviation of study hours for each level of importance.

(c) Comment briefly on how the groups compare in terms of study hours.

(d) Make a suitable graph of this data set.

(e) The statistician in this study decided that the data were sufficiently normal in shape given the (very large) sample sizes, but was concerned about unequal spreads among the three groups. Given this, run a suitable analysis and display the output. (This includes a suitable follow-up test, if warranted.)

(f) What do you conclude from your analysis of the previous part, in the context of the data?

My solutions follow:

[2] Gaelic is a language of Scotland and Ireland, and the culture of the people who speak it.

11.10 Movie ratings and lengths

Before a movie is shown in theatres, it receives a "rating" that says what kind of material it contains. link explains the categories, from G (suitable for children) to R (anyone under 17 must be accompanied by parent/guardian). In 2011, two students collected data on the length (in minutes) and the rating category, for 15 movies of each rating category, randomly chosen from all the movies released that year. The data are at link.

(a) Read the data into R, and display (some of) what you read in.

Solution

`read_csv`:

```
my_url <- "http://ritsokiguess.site/datafiles/movie-lengths.csv"
movies <- read_csv(my_url)
```

```
Rows: 60 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): rating
dbl (1): length

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
movies
```

```
# A tibble: 60 x 2
   length rating
    <dbl> <chr>
 1     25 G
 2     75 G
 3     88 G
 4     63 G
 5     76 G
 6     97 G
 7     68 G
 8     82 G
 9     98 G
10     74 G
# i 50 more rows
```

Something that looks like a length in minutes, and a rating.

■

(b) Count how many movies there are of each rating.

Solution

```
movies %>% count(rating)
```

```
# A tibble: 4 x 2
  rating     n
  <chr>  <int>
1 G         15
2 PG        15
3 PG-13     15
4 R         15
```

Fifteen of each rating. (It's common to have the same number of observations in each group, but not necessary for a one-way ANOVA.)

■

(c) Carry out an ANOVA and a Tukey analysis (if warranted).

Solution

ANOVA first:

```
length.1 <- aov(length ~ rating, data = movies)
summary(length.1)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
rating       3  14624    4875   11.72 4.59e-06 ***
Residuals   56  23295     416
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This P-value is 0.00000459, which is way less than 0.05.

Having rejected the null (which said "all means equal"), we now need to do Tukey, thus:

```
TukeyHSD(length.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = length ~ rating, data = movies)

$rating
               diff        lwr       upr     p adj
PG-G      26.333333   6.613562 46.053104 0.0044541
PG-13-G   42.800000  23.080229 62.519771 0.0000023
R-G       30.600000  10.880229 50.319771 0.0007379
PG-13-PG  16.466667  -3.253104 36.186438 0.1327466
R-PG       4.266667 -15.453104 23.986438 0.9397550
R-PG-13  -12.200000 -31.919771  7.519771 0.3660019
```

Cast your eye down the `p adj` column and look for the ones that are significant, here the first three. These are all comparisons with the G ("general") movies, which are shorter on average than the others (which are not significantly different from each other).

If you like, you can make a table of means to verify that:

```r
movies %>%
  group_by(rating) %>%
  summarize(mean = mean(length))
```

```
# A tibble: 4 x 2
  rating  mean
  <chr>  <dbl>
1 G       80.6
2 PG     107.
3 PG-13  123.
4 R      111.
```

When we do this problem in SAS, you'll see the Tukey get handled a different way, one that you might find more appealing.

■

(d) Make a graph to assess whether this ANOVA is trustworthy. Discuss your graph and its implications briefly.

Solution

The obvious graph is a boxplot:

```r
ggplot(movies, aes(x = rating, y = length)) + geom_boxplot()
```

For ANOVA, we are looking for approximately normal distributions within each group and approximately equal spreads. Without the outliers, I would be more or less happy with that, but the G movies have a low outlier that would pull the mean down and the PG and PG-13 movies have outliers that would pull the mean up. So a comparison of means might make the differences look more significant than they should. Having said that, you could also say that the ANOVA is *very* significant, so even considering the effect of the outliers, the differences between G and the others are still likely to be significant.

Extra: the way to go if you don't trust the ANOVA is (as for the two-sample t) the Mood's median test. This applies to any number of groups, and works in the same way as before:

```r
library(smmr)
median_test(movies, length, rating)
```

```
$table
       above
group   above below
  G         2    13
  PG        7     7
  PG-13    12     3
```

```
  R         8     6

$test
       what        value
1 statistic 13.752380952
2        df  3.000000000
3   P-value  0.003262334
```

Still significant, though not quite as small a P-value as before (which echoes our thoughts about what the outliers might do to the means). If you look at the table above the test results, you see that the G movies are mostly shorter than the overall median, but now the PG-13 movies are mostly *longer*. So the picture is a little different.

Mood's median test does not naturally come with something like Tukey. What you can do is to do all the pairwise Mood's median tests, between each pair of groups, and then adjust to allow for your having done several tests at once. I thought this was generally useful enough that I put it into `smmr` under the name `pairwise_median_test`:

```r
pairwise_median_test(movies, length, rating)
```

```
# A tibble: 6 x 4
  g1    g2      p_value adj_p_value
  <chr> <chr>     <dbl>       <dbl>
1 G     PG    0.00799      0.0479
2 G     PG-13 0.0000590    0.000354
3 G     R     0.0106       0.0635
4 PG    PG-13 0.0106       0.0635
5 PG    R     0.715        1
6 PG-13 R     0.273        1
```

You can ignore those (adjusted) P-values rather stupidly bigger than 1. These are not significant.

There are two significant differences in median length: between G movies and the two flavours of PG movies. The G movies are significantly shorter (as you can tell from the boxplot), but the difference between G and R movies is no longer significant (a change from the regular ANOVA).

You may be puzzled by something in the boxplot: how is it that the G movies are significantly shorter than the PG movies, but not significantly shorter than the R movies, *when the difference in medians between G and R movies is bigger*? In Tukey, if the difference in

means is bigger, the P-value is smaller.[3] The resolution to this puzzle, such as it is, is that Mood's median test is not directly comparing the medians of the groups (despite its name); it's counting values above and below a *joint* median, which might be a different story.

■

11.11 Deer and how much they eat

Do adult deer eat different amounts of food at different times of the year? The data in link are the weights of food (in kilograms) consumed by randomly selected adult deer observed at different times of the year (in February, May, August and November). We will assume that these were different deer observed in the different months. (If the same animals had been observed at different times, we would have been in the domain of "repeated measures", which would require a different analysis, beyond the scope of this course.)

(a) Read the data into R, and calculate numbers of observations and the median amounts of food eaten each month.

Solution

The usual stuff for data values separated by (single) spaces:

```
myurl <- "http://ritsokiguess.site/datafiles/deer.txt"
deer <- read_delim(myurl, " ")
```

```
Rows: 22 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): month
dbl (1): food

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

and then, recalling that `n()` is the handy way of getting the number of observations in each group:

[3] Actually, this doesn't always work if the sample sizes in each group are different. If you're comparing two small groups, it takes a *very large* difference in means to get a small P-value. But in this case the sample sizes are all the same.

```r
deer %>%
  group_by(month) %>%
  summarize(n = n(), med = median(food))
```

```
# A tibble: 4 x 3
  month     n   med
  <chr> <int> <dbl>
1 Aug       6  4.7
2 Feb       5  4.8
3 May       6  4.35
4 Nov       5  5.2
```

When you want the number of observations *plus* some other summaries, as here, the group-by and summarize idea is the way, using `n()` to get the number of observations in each group. `count` counts the number of observations per group when you *only* have grouping variables.

The medians differ a bit, but it's hard to judge without a sense of spread, which the boxplots (next) provide. November is a bit higher and May a bit lower.

■

(b) Make side-by-side boxplots of the amount of food eaten each month. Comment briefly on what you see.

Solution

```r
ggplot(deer, aes(x = month, y = food)) + geom_boxplot()
```

This offers the suggestion that maybe November will be significantly higher than the rest and May significantly lower, or at least they will be significantly different from each other.

This is perhaps getting ahead of the game: we should be thinking about spread and shape. Bear in mind that there are only 5 or 6 observations in each group, so you won't be able to say much about normality. In any case, we are going to be doing a Mood's median test, so any lack of normality doesn't matter (eg. perhaps that 4.4 observation in August). Given the small sample sizes, I actually think the spreads are quite similar.

Another way of looking at the data, especially with these small sample sizes, is a "dot plot": instead of making a boxplot for each month, we plot the actual points for each month as if we were making a scatterplot:

```
ggplot(deer, aes(x = month, y = food)) + geom_point()
```

Wait a minute. There were five deer in February and six in August. Where did they go?

The problem is *overplotting*: more than one of the deer plotted in the same place on the plot, because the amounts of food eaten were only given to one decimal place and there were some duplicated values. One way to solve this is to randomly move the points around so that no two of them plot in the same place. This is called *jittering*, and is done like this:

```
ggplot(deer, aes(x = month, y = food)) + geom_jitter(width = 0, height = 0.05)
```

Now you see all the deer, and you can see that two pairs of points in August and one pair of points in February are close enough on the jittered plot that they would have been the same to one decimal place.

I wanted to keep the points above the months they belong to, so I only allowed vertical jitter (that's the `width` and `height` in the `geom_jitter`; the width is zero so there is no horizontal jittering). If you like, you can colour the months; it's up to you whether you think that's making the plot easier to read, or is overkill (see my point on the facetted plots on the 2017 midterm).

This way you see the whole distribution for each month. Normally it's nicer to see the summary made by the boxplots, but here there are not very many points. The value of 4.4 in August does look quite a bit lower than the rest, but the other months look believably normal given the small sample sizes. I don't know about equal spreads (November looks more spread out), but normality looks believable. Maybe this is the kind of situation in which Welch's ANOVA is a good idea. (If you believe that the normality-with-unequal-spreads is a reasonable assumption to make, then the Welch ANOVA will be more powerful than the Mood's median test, and so should be preferred.)

■

(c) Run a Mood's median test as in lecture (ie. not using `smmr`). What do you conclude, in the context of the data?

Solution

To give you some practice with the mechanics, first find the overall median:

```
deer %>% summarize(med = median(food))
```

```
# A tibble: 1 x 1
    med
  <dbl>
1   4.7
```

or

```
median(deer$food)
```

```
[1] 4.7
```

I like the first way because it's the same idea as we did before, just not differentiating by month. I think there are some observations exactly equal to the median, which will mess things up later:

```
deer %>% filter(food == 4.7)
```

```
# A tibble: 4 x 2
  month  food
  <chr> <dbl>
1 Feb     4.7
2 Feb     4.7
3 Aug     4.7
4 Aug     4.7
```

There are, two in February and two in August.

Next, make (and save) a table of the observations within each month that are above and below this median:

```
tab1 <- with(deer, table(month, food < 4.7))
tab1
```

```
month FALSE TRUE
  Aug     4    2
  Feb     5    0
  May     0    6
  Nov     5    0
```

or

```r
tab2 <- with(deer, table(month, food > 4.7))
tab2
```

```
month FALSE TRUE
  Aug     4    2
  Feb     2    3
  May     6    0
  Nov     0    5
```

Either of these is good, but note that they are different. Two of the February observations (the ones that were exactly 4.7) have "switched sides", and (look carefully) two of the August ones also. Hence the test results will be different, and `smmr` (later) will give different results again:

```r
chisq.test(tab1, correct = F)
```

```
Warning in chisq.test(tab1, correct = F): Chi-squared approximation may be
incorrect


    Pearson's Chi-squared test

data:  tab1
X-squared = 16.238, df = 3, p-value = 0.001013
```

```r
chisq.test(tab2, correct = F)
```

```
Warning in chisq.test(tab2, correct = F): Chi-squared approximation may be
incorrect
```

```
    Pearson's Chi-squared test

data:  tab2
X-squared = 11.782, df = 3, p-value = 0.008168
```

The warnings are because of the small frequencies. If you've done these by hand before (which you will have if you took PSYC08), you'll remember that thing about "expected frequencies less than 5". This is that. It means "don't take those P-values *too* seriously."

The P-values are different, but they are both clearly significant, so the median amounts of food eaten in the different months are not all the same. (This is the same "there are differences" that you get from an ANOVA, which you would follow up with Tukey.) Despite the injunction not to take the P-values too seriously, I think these are small enough that they could be off by a bit without affecting the conclusion.

The first table came out with a smaller P-value because it looked more extreme: all of the February measurements were taken as higher than the overall median (since we were counting "strictly less" and "the rest"). In the second table, the February measurements look more evenly split, so the overall P-value is not quite so small.

You can make a guess as to what `smmr` will come out with (next), since it throws away any data values exactly equal to the median.

■

(d) Run a Mood's median test using `smmr`, and compare the results with the previous part.

Solution

Off we go:

```r
library(smmr)
median_test(deer, food, month)
```

```
$table
     above
group above below
  Aug     2     2
  Feb     3     0
  May     0     6
  Nov     5     0

$test
       what        value
```

```
1 statistic 13.950000000
2        df  3.000000000
3   P-value  0.002974007
```

The P-value came out in between the other two, but the conclusion is the same all three ways: the months are not all the same in terms of median food eaten. The researchers can then go ahead and try to figure out *why* the animals eat different amounts in the different months.

You might be wondering how you could get rid of the equal-to-median values in the build-it-yourself way. This is `filter` from `dplyr`, which you use first:

```
deer2 <- deer %>% filter(food != 4.7)
tab3 <- with(deer2, table(month, food < 4.7))
tab3
```

```
month FALSE TRUE
  Aug     2    2
  Feb     3    0
  May     0    6
  Nov     5    0
```

```
chisq.test(tab3)
```

```
Warning in chisq.test(tab3): Chi-squared approximation may be incorrect


    Pearson's Chi-squared test

data:  tab3
X-squared = 13.95, df = 3, p-value = 0.002974
```

which is exactly what `smmr` does, so the answer is identical.[4] How would an ANOVA come out here? My guess is, very similarly:

[4]The computer scientists among you will note that I should not use equals or not-equals to compare a decimal floating-point number, since decimal numbers are not represented exactly in the computer. R, however, is ahead of us here, since when you try to do "food not equal to 4.7", it tests whether food is more than a small distance away from 4.7, which is the right way to do it. In R, therefore, code like my `food != 4.7` does exactly what I want, but in a language like C, it *does not*, and you have to be more careful: `abs(food-4.7)>1e-8`, or something like that. The small number `1e-8` (10^{-8}) is typically equal to **machine epsilon**, the smallest number on a computer that is distinguishable from zero.

```r
deer.1 <- aov(food ~ month, data = deer)
summary(deer.1)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)    
month        3 2.3065  0.7688   22.08 2.94e-06 ***
Residuals   18 0.6267  0.0348                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
TukeyHSD(deer.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = food ~ month, data = deer)

$month
              diff         lwr        upr     p adj
Feb-Aug  0.1533333 -0.16599282  0.4726595 0.5405724
May-Aug -0.3333333 -0.63779887 -0.0288678 0.0290758
Nov-Aug  0.5733333  0.25400718  0.8926595 0.0004209
May-Feb -0.4866667 -0.80599282 -0.1673405 0.0021859
Nov-Feb  0.4200000  0.08647471  0.7535253 0.0109631
Nov-May  0.9066667  0.58734052  1.2259928 0.0000013
```

The conclusion is the same, but the P-value on the F-test is much smaller. I think this is because the F-test uses the actual values, rather than just whether they are bigger or smaller than 4.7. The Tukey says that all the months are different in terms of (now) mean, except for February and August, which were those two very similar ones on the boxplot.

■

(e) How is it that Mood's median test does not completely answer the question you really want to answer? How might you get an answer to the question you *really* want answered? Explain briefly, and obtain the answer you *really* want, discussing your results briefly.

Solution

That's rather a lot, so let's take those things one at a time.[5]

[5]Most of these parts are old from assignment questions that I actually asked a previous class to do, but not this part. I added it later.

Mood's median test is really like the F-test in ANOVA: it's testing the null hypothesis that the groups (months) all have the same median (of food eaten), against the alternative that the null is not true. We rejected this null, but we don't know which months differ significantly from which. To resolve this in ANOVA, we do Tukey (or Games-Howell if we did the Welch ANOVA). The corresponding thing here is to do all the possible two-group Mood tests on all the pairs of groups, and, after adjusting for doing (here) six tests at once, look at the adjusted P-values to see how the months differ in terms of food eaten.

This is accomplished in `smmr` via `pairwise_median_test`, thus:

```
pairwise_median_test(deer, food, month)
```

```
# A tibble: 6 x 4
  g1    g2    p_value adj_p_value
  <chr> <chr>   <dbl>       <dbl>
1 Aug   Feb   0.147       0.884
2 Aug   May   0.0209      0.126
3 Aug   Nov   0.00270     0.0162
4 Feb   May   0.00157     0.00939
5 Feb   Nov   0.0578      0.347
6 May   Nov   0.00157     0.00939
```

This compares each month with each other month. Looking at the last column, there are only three significant differences: August-November, February-May and May-November. Going back to the table of medians we made in (a), November is significantly higher (in terms of median food eaten) than August and May (but not February), and February is significantly higher than May. The other differences are not big enough to be significant.

Extra: Pairwise median tests done this way are not likely to be very sensitive (that is, powerful), for a couple of reasons: (i) the usual one that the median tests don't use the data very efficiently, and (ii) the way I go from the unadjusted to the adjusted P-values is via Bonferroni (here, multiply the P-values by 6), which is known to be safe but conservative. This is why the Tukey produced more significant differences among the months than the pairwise median tests did.

■

11.12 Movie ratings again

This question again uses the movie rating data at link.

(a) Read the data into R and obtain the number of movies of each rating and the *median* length of movies of each rating.

Solution

Reading in is as in the other question using these data (just copy your code, or mine).

```
my_url <- "http://ritsokiguess.site/datafiles/movie-lengths.csv"
movies <- read_csv(my_url)
```

```
Rows: 60 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): rating
dbl (1): length

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
movies
```

```
# A tibble: 60 x 2
   length rating
    <dbl> <chr>
 1     25 G
 2     75 G
 3     88 G
 4     63 G
 5     76 G
 6     97 G
 7     68 G
 8     82 G
 9     98 G
10     74 G
# i 50 more rows
```

Now, the actual for-credit part, which is a `group_by` and `summarize`:

```
movies %>%
  group_by(rating) %>%
  summarize(count = n(), med = median(length))
```

```
# A tibble: 4 x 3
  rating count   med
  <chr>  <int> <dbl>
1 G         15    82
2 PG        15   100
3 PG-13     15   117
4 R         15   103
```

The G movies have a smaller median than the others, but also the PG-13 movies seem to be longer on average (not what we found before).

■

(b) Obtain a suitable graph that assesses the assumptions for ANOVA. Why do you think it is not reasonable to run ANOVA here? Explain briefly.

Solution

The graph would seem to be a boxplot, side by side for each group:

```
ggplot(movies, aes(x = rating, y = length)) + geom_boxplot()
```

We are looking for approximate normal distributions with approximately equal spreads, which I don't think we have: there are outliers, at the low end for G movies, and at the high end for

PG and PG-13 movies. Also, you might observe that the distribution of lengths for R movies is skewed to the right. (Noting either the outliers or skewness as a reason for not believing normality is enough, since all we need is *one* way that normality fails.)

I think the spreads (as measured by the interquartile ranges) are acceptably similar, but since we have rejected normality, it is a bit late for that.

So I think it is far from reasonable to run an ANOVA here. In my opinion 15 observations in each group is not enough to gain much from the Central Limit Theorem either.

Extra: since part of the assumption for ANOVA is (approximate) normality, it would also be entirely reasonable to make normal quantile plots, one for each movie type, facetted. Remember the process: you pretend that you are making a normal quantile plot for all the data together, regardless of group, and then at the last minute, you throw in a `facet_wrap`. I've written the code out on three lines, so that you can see the pieces: the "what to plot", then the normal quantile plot part, then the facetting:

```
ggplot(movies, aes(sample = length)) +
  stat_qq() + stat_qq_line() +
  facet_wrap(~rating)
```

Since there are four movie ratings, `facet_wrap` has arranged them into a 2×2 grid, which satisfyingly means that each normal quantile plot is more or less square and thus easy to interpret.

The principal problem unveiled by these plots is outliers. It looks as if the G movies have one low outlier, the PG movies have two high outliers, the PG-13 movies have one or maybe three high outliers (depending on how you count them), and the R movies have none. Another way to look at the last two is you could call them curved, with too much bunching up at the bottom and (on PG-13) too much spread-out-ness at the top, indicating right-skewed distributions. The distribution of lengths of the R-rated movies is too bunched up at the bottom, but as you would expect for a normal at the top. The R movies show the right-skewedness in an odd way: usually this skewness shows up by having too many high values, but this time it's having too *few low* values.

The assumption for ANOVA is that all four of these are at least approximately normal (with the same spread). We found problems with the normality on at least three of them, so we definitely have doubts about trusting ANOVA here.

I could have used `scales=free` here to get a separate y-axis for each plot, but since the y-axis is movie length each time, and all four groups would be expected to have at least roughly similar movie lengths, I left it as it was. (The other advantage of leaving the scales the same is that you can compare spread by comparing the slopes of the lines on these graphs; since the lines connect the observed and theoretical quartiles, a steeper slope means a larger IQR. Here, the R line is steepest and the PG line is flattest. Compare this with the spreads of the boxplots.)

Extra extra: if you want, you can compare the normal quantile plots with the boxplots to see whether you get the same conclusion from both. For the G movies, the low outlier shows up both ways, and the rest of the distribution is at least more or less normal. For the PG movies, I'd say the distribution is basically normal except for the highest two values (on both plots). For the PG-13 movies, only the highest value shows up as an outlier, but the next two apparent outliers on the normal quantile plot are at the upper end of the long upper whisker, so the boxplot is saying "right-skewed with one upper outlier" rather than "three upper outliers". The distribution of the R movies is skewed right, with the bunching at the bottom showing up as the very small lower whisker.

The boxplots and the normal quantile plots are basically telling the same story in each case, but they are doing it in a slightly different way.

■

(c) Run a Mood's median test (use `smmr` if you like). What do you conclude, in the context of the data?

Solution

The smart way is to use `smmr`, since it is much easier:

```r
library(smmr)
median_test(movies, length, rating)
```

```
$table
       above
group   above below
  G         2    13
  PG        7     7
  PG-13    12     3
  R         8     6

$test
       what        value
1 statistic 13.752380952
2        df  3.000000000
3   P-value  0.003262334
```

The movies do not all have the same median length, or at least one of the rating types has movies of different median length from the others. Or something equivalent to that. It's the same conclusion as for ANOVA, only with medians instead of means.

You can speculate about why the test came out significant. My guess is that the G movies are shorter than average, and that the PG-13 movies are longer than average. (We had the first conclusion before, but not the second. This is where medians are different from means.)

The easiest way to see which movie types really differ in length from which is to do all the pairwise median tests, which is in `smmr` thus:

```r
pairwise_median_test(movies, length, rating)
```

```
# A tibble: 6 x 4
  g1    g2      p_value adj_p_value
  <chr> <chr>     <dbl>       <dbl>
1 G     PG    0.00799      0.0479
2 G     PG-13 0.0000590    0.000354
3 G     R     0.0106       0.0635
4 PG    PG-13 0.0106       0.0635
5 PG    R     0.715        1
6 PG-13 R     0.273        1
```

The inputs for this are the same ones in the same order as for `median_test`. (A design decision on my part, since otherwise *I* would never have been able to remember how to run these!) Only

the first two of these are significant (look in the last column). We can remind ourselves of the sample medians:

```
movies %>%
  group_by(rating) %>%
  summarize(count = n(), med = median(length))
```

```
# A tibble: 4 x 3
  rating count   med
  <chr>  <int> <dbl>
1 G         15    82
2 PG        15   100
3 PG-13     15   117
4 R         15   103
```

The G movies are significantly shorter than the PG and PG-13 movies, but not quite significantly different from the R movies. This is a little odd, since the difference in sample medians between G and PG, significant, is *less* than for G and R (not significant). There are several Extras here, which you can skip if you don't care about the background. First, we can do the median test by hand: This has about four steps: (i) find the median of all the data, (ii) make a table tabulating the number of values above and below the overall median for each group, (iii) test the table for association, (iv) draw a conclusion. Thus (i):

```
median(movies$length)
```

```
[1] 100
```

or

```
movies %>% summarize(med = median(length))
```

```
# A tibble: 1 x 1
    med
  <dbl>
1   100
```

or store it in a variable, and then (ii):

```r
tab1 <- with(movies, table(length < 100, rating))
tab1
```

```
       rating
         G PG PG-13  R
  FALSE  2  8    12  9
  TRUE  13  7     3  6
```

or

```r
tab2 <- with(movies, table(length > 100, rating))
tab2
```

```
       rating
         G PG PG-13  R
  FALSE 13  8     3  7
  TRUE   2  7    12  8
```

These differ because there are evidently some movies of length exactly 100 minutes, and it matters whether you count $<$ and $\geq$ (as in `tab1`) or $>$ and *le* (`tab2`). Either is good.

Was I right about movies of length exactly 100 minutes?

```r
movies %>% filter(length == 100)
```

```
# A tibble: 2 x 2
  length rating
   <dbl> <chr>
1    100 PG
2    100 R
```

One PG and one R. It makes a difference to the R movies, but if you look carefully, it makes a difference to the PG movies as well, because the False and True switch roles between `tab1` and `tab2` (compare the G movies, for instance). You need to store your table in a variable because it has to get passed on to `chisq.test` below, (iii):

```r
chisq.test(tab1, correct = FALSE)
```

```
    Pearson's Chi-squared test

data:  tab1
X-squared = 14.082, df = 3, p-value = 0.002795
```

or

```
chisq.test(tab2, correct = FALSE)
```

```
    Pearson's Chi-squared test

data:  tab2
X-squared = 13.548, df = 3, p-value = 0.003589
```

Either is correct, or, actually, without the `correct=FALSE`.[6]

The conclusion (iv) is the same either way: the null of no association is clearly rejected (with a P-value of 0.0028 or 0.0036 as appropriate), and therefore whether a movie is longer or shorter than median length depends on what rating it has: that is, the median lengths do differ among the ratings. The same conclusion, in other words, as the F-test gave, though with not quite such a small P-value.

Second, you might be curious about how we might do something like Tukey having found some significant differences (that is, what's lurking in the background of `pairwise_median_test`).

Let's first suppose we are comparing G and PG movies. We need to pull out just those, and then compare them using `smmr`. Because the first input to `median_test` is a data frame, it fits neatly into a pipe (with the data frame omitted):

```
movies %>%
  filter(rating == "G" | rating == "PG") %>%
  median_test(length, rating)
```

```
$table
     above
group above below
   G      4    11
   PG    10     3
```

[6]See discussion elsewhere about Yates' Correction and fixed margins.

```
$test
       what       value
1 statistic 7.035897436
2        df 1.000000000
3   P-value 0.007989183
```

We're going to be doing this about six times — $\binom{4}{2} = 6$ choices of two rating groups to compare out of the four — so we should have a function to do it. I think the input to the function should be a data frame that has a column called `rating`, and two names of ratings to compare:

```
comp2 <- function(rat_1, rat_2, d) {
  d %>%
    filter(rating == rat_1 | rating == rat_2) %>%
    median_test(length, rating)
}
```

The way I wrote this function is that you have to specify the movie ratings in quotes. It is *possible* to write it in such a way that you input them without quotes, `tidyverse` style, but that gets into "non-standard evaluation" and `enquo()` and `!!`, which (i) I have to look up every time I want to do it, and (ii) I am feeling that the effort involved in explaining it to you is going to exceed the benefit you will gain from it. I mastered it enough to make it work in `smmr` (note that you specify column names without quotes there). There are tutorials on this kind of thing if you're interested.

Anyway, testing:

```
comp2("G", "PG", movies)
```

```
$table
     above
group above below
   G      4    11
   PG    10     3

$test
       what       value
1 statistic 7.035897436
2        df 1.000000000
3   P-value 0.007989183
```

That works, but I really only want to pick out the P-value, which is in the list item `test` in the column `value`, the third entry. So let's rewrite the function to return just that:

```
comp2 <- function(rat_1, rat_2, d) {
  d %>%
    filter(rating == rat_1 | rating == rat_2) %>%
    median_test(length, rating) %>%
    pluck("test", "value", 3)
}
comp2("G", "PG", movies)
```

```
[1] 0.007989183
```

Gosh.

What `median_test` returns is an R `list` that has two things in it, one called `table` and one called `test`. The thing called `test` is a data frame with a column called `value` that contains the P-values. The third of these is the two-sided P-value that we want.

You might not have seen `pluck` before. This is a way of getting things out of complicated data structures. This one takes the output from `median_test` and from it grabs the piece called `test`. This is a data frame. Next, we want the column called `value`, and from that we want the third row. These are specified one after the other to `pluck` and it pulls out the right thing.

So now our function returns just the P-value.

I have to say that it took me several goes and some playing around in R Studio to sort this one out. Once I thought I understood `pluck`, I wondered why my function was not returning a value. And then I realized that I was saving the value inside the function and not returning it. Ooops. The nice thing about `pluck` is that I can put it on the end of the pipeline and and it will pull out (and return) whatever I want it to.

Let's grab a hold of the different rating groups we have:

```
the_ratings <- unique(movies$rating)
the_ratings
```

```
[1] "G"     "PG-13" "PG"    "R"
```

The Pythonisti among you will know how to finish this off: do a loop-inside-a-loop over the rating groups, and get the P-value for each pair. You can do that in R, if you must. It's not pretty at all, but it works:

```r
ii <- character(0)
jj <- character(0)
pp <- numeric(0)
for (i in the_ratings) {
  for (j in the_ratings) {
    pval <- comp2(i, j, movies)
    ii <- c(ii, i)
    jj <- c(jj, j)
    pp <- c(pp, pval)
  }
}
tibble(ii, jj, pp)
```

```
# A tibble: 16 x 3
   ii    jj           pp
   <chr> <chr>     <dbl>
 1 G     G     1
 2 G     PG-13 0.0000590
 3 G     PG    0.00799
 4 G     R     0.0106
 5 PG-13 G     0.0000590
 6 PG-13 PG-13 1
 7 PG-13 PG    0.0106
 8 PG-13 R     0.273
 9 PG    G     0.00799
10 PG    PG-13 0.0106
11 PG    PG    1
12 PG    R     0.715
13 R     G     0.0106
14 R     PG-13 0.273
15 R     PG    0.715
16 R     R     1
```

This is a lot of fiddling about, since you have to initialize three vectors, and then update them every time through the loop. It's hard to read, because the actual business part of the loop is the calculation of the P-value, and that's almost hidden by all the book-keeping. (It's also slow and inefficient, though the slowness doesn't matter too much here since it's not a very big problem.)

Let's try another way:

```r
crossing(first = the_ratings, second = the_ratings)
```

```
# A tibble: 16 x 2
   first second
   <chr> <chr>
 1 G     G
 2 G     PG
 3 G     PG-13
 4 G     R
 5 PG    G
 6 PG    PG
 7 PG    PG-13
 8 PG    R
 9 PG-13 G
10 PG-13 PG
11 PG-13 PG-13
12 PG-13 R
13 R     G
14 R     PG
15 R     PG-13
16 R     R
```

This does "all possible combinations" of one rating with another. We don't actually need all of that; we just need the ones where the first one is (alphabetically) strictly less than the second one. This is because we're never comparing a rating with itself, and each pair of ratings appears twice, once in alphabetical order, and once the other way around. The ones we need are these:

```r
crossing(first = the_ratings, second = the_ratings) %>%
  filter(first < second)
```

```
# A tibble: 6 x 2
  first second
  <chr> <chr>
1 G     PG
2 G     PG-13
3 G     R
4 PG    PG-13
5 PG    R
6 PG-13 R
```

A technique thing to note: instead of asking "how do I pick out the distinct pairs of ratings?", I use two simpler tools: first I make all the combinations of pairs of ratings, and then out of those, pick the ones that are alphabetically in ascending order, which we know how to do.

Now we want to call our function `comp2` for each of the things in `first` *and* each of the things in `second`, and make a new column called `pval` that contains exactly that. `comp2` expects single movie ratings for each of its inputs, not a vector of each, so the way to go about this is `rowwise`:

```
crossing(first = the_ratings, second = the_ratings) %>%
  filter(first < second) %>%
  rowwise() %>%
  mutate(pval = comp2(first, second, movies))
```

```
# A tibble: 6 x 3
# Rowwise:
  first second      pval
  <chr> <chr>      <dbl>
1 G     PG     0.00799
2 G     PG-13  0.0000590
3 G     R      0.0106
4 PG    PG-13  0.0106
5 PG    R      0.715
6 PG-13 R      0.273
```

One more thing: we're doing 6 tests at once here, so we're giving ourselves 6 chances to reject a null (all medians equal) that might have been true. So the true probability of a type I error is no longer 0.05 but something bigger.

The easiest way around that is to do a so-called Bonferroni adjustment: instead of rejecting if the P-value is less than 0.05, we only reject if it is less than 0.05/6, since we are doing 6 tests. This is a fiddly calculation to do by hand, but it's easy to build in another `mutate`, thus:[7]

```
crossing(first = the_ratings, second = the_ratings) %>%
  filter(first < second) %>%
  rowwise() %>%
  mutate(pval = comp2(first, second, movies)) %>%
  mutate(reject = (pval < 0.05 / 6))
```

[7] In the pairwise median test in `smmr`, I did this backwards: rather than changing the alpha that you compare each P-value with from 0.05 to 0.05/6, I flip it around so that you adjust the P-values by *multiplying* them by 6, and then comparing the adjusted P-values with the usual 0.05. It comes to the same place in the end, except that this way you can get adjusted P-values that are greater than 1, which makes no sense. You read those as being definitely not significant.

```
# A tibble: 6 x 4
# Rowwise:
  first second      pval reject
  <chr> <chr>      <dbl> <lgl>
1 G     PG     0.00799   TRUE
2 G     PG-13  0.0000590 TRUE
3 G     R      0.0106    FALSE
4 PG    PG-13  0.0106    FALSE
5 PG    R      0.715     FALSE
6 PG-13 R      0.273     FALSE
```

And not a loop in sight.

This is how I coded it in `pairwise_median_test`. If you want to check it, it's on Github: link. The function `median_test_pair` is the same as `comp2` above.

So the only significant differences are now G compared to PG and PG-13. There is not a significant difference in median movie length between G and R, though it is a close call. We thought the PG-13 movies might have a significantly different median from other rating groups beyond G, but they turn out not to have. (The third and fourth comparisons would have been significant had we not made the Bonferroni adjustment to compensate for doing six tests at once; with that adjustment, we only reject if the P-value is less than $0.05/6 = 0.0083$, and so 0.0106 is not quite small enough to reject with.)

Listing the rating groups sorted by median would give you an idea of how far different the medians have to be to be significantly different:

```r
medians <- movies %>%
  group_by(rating) %>%
  summarize(med = median(length)) %>%
  arrange(desc(med))
medians
```

```
# A tibble: 4 x 2
  rating   med
  <chr>  <dbl>
1 PG-13    117
2 R        103
3 PG       100
4 G         82
```

Something rather interesting has happened: even though the comparison of G and PG (18 apart) is significant, the comparison of G and R (21 apart) is not significant. This seems very

odd, but it happens because the Mood median test is not actually literally comparing the sample medians, but only assessing the splits of values above and below the median of the combined sample. A subtlety, rather than an error, I'd say.

Here's something extremely flashy to finish with:

```
crossing(first = the_ratings, second = the_ratings) %>%
  filter(first < second) %>%
  rowwise() %>%
  mutate(pval = comp2(first, second, movies)) %>%
  mutate(reject = (pval < 0.05 / 6)) %>%
  left_join(medians, by = c("first" = "rating")) %>%
  left_join(medians, by = c("second" = "rating"))
```

```
# A tibble: 6 x 6
# Rowwise:
  first second      pval reject med.x med.y
  <chr> <chr>      <dbl> <lgl>  <dbl> <dbl>
1 G     PG     0.00799   TRUE      82   100
2 G     PG-13  0.0000590 TRUE      82   117
3 G     R      0.0106    FALSE     82   103
4 PG    PG-13  0.0106    FALSE    100   117
5 PG    R      0.715     FALSE    100   103
6 PG-13 R      0.273     FALSE    117   103
```

The additional two lines look up the medians of the rating groups in `first`, then `second`, so that you can see the actual medians of the groups being compared each time. You see that medians different by 30 are definitely different, ones differing by 15 or less are definitely not different, and ones differing by about 20 could go either way.

I think that's *quite* enough of that.

■

11.13 Atomic weight of carbon

The atomic weight of the chemical element carbon is 12. Two methods of measuring the atomic weight of samples of carbon were compared. The results are shown in link. The methods are labelled 1 and 2. The first task is to find out whether the two methods have different "typical" measures (mean or median, as appropriate) of the atomic weight of carbon.

For this question, compose a report in R Markdown. (R Markdown is what you use in an R Notebook, but you can also have a separate R Markdown document from which you can produce HTML, Word etc. output.) See part (a) for how to get this started.

Your report should read like an actual report, not just the answers to some questions that I set you. To help with that, write some text that links the parts of the report together smoothly, so that it reads as a coherent whole. The grader had 3 discretionary marks to award for the overall quality of your writing. The scale for this was:

- 3 points: excellent writing. The report flows smoothly, is easy to read, and contains everything it should (and nothing it shouldn't).
- 2 points: satisfactory writing. Not the easiest to read, but says what it should, and it looks at least somewhat like a report rather than a string of answers to questions.
- 1 point: writing that is hard to read or to understand. If you get this (or 0), you should consider what you need to do to improve when you write your project.
- 0 points: you answered the questions, but you did almost nothing to make it read like a report.

(a) Create a new R Markdown document. To do this, in R Studio, select File, New File, R Markdown. Type the report title and your name in the boxes, and leave the output on the default HTML. Click OK.

Solution

You'll see the title and your name in a section at the top of the document, and below that you'll see a template document, as you would for an R Notebook. The difference is that where you are used to seeing Preview, it now says "knit", but this has the same effect of producing the formatted version of your report.

■

(b) Write an introduction that explains the purpose of this study and the data collected in your own words.

Solution

Something like this:

> This study is intended to compare two different methods (labelled 1 and 2) for measuring the atomic weight of carbon (which is known in actual fact to be 12). Fifteen samples of carbon were used; ten of these were assessed using method 1 and the remaining five using method 2. The primary interest in this particular study is to see whether there is a difference in the mean or median atomic weight as measured by the two methods.

Before that, start a new section like this: `## Introduction`. Also, get used to expressing your understanding in your words, not mine. Using my words, in my courses, is likely to be worth very little.

■

(c) Begin an appropriately-titled new section in your report, read the data into R and display the results.

Solution

Values separated by spaces:

```
my_url <- "http://ritsokiguess.site/datafiles/carbon.txt"
carbon <- read_delim(my_url, " ")
```

```
Rows: 15 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): method, weight

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
carbon
```

```
# A tibble: 15 x 2
   method weight
    <dbl>  <dbl>
 1      1   12.0
 2      1   12.0
 3      1   12.0
 4      1   12.0
 5      1   12.0
 6      1   12.0
 7      1   12.0
 8      1   12.0
 9      1   12.0
10      1   12.0
11      2   12.0
12      2   12.0
13      2   12.0
```

```
14       2   12.0
15       2   12.0
```

I would expect you to include, without being told to include it, some text in your report indicating that you have sensible data: two methods labelled 1 and 2 as promised, and a bunch[8] of atomic weights close to the nominal figure of 12.

■

(d) Make an appropriate plot to compare the measurements obtained by the two methods. You might need to do something about the two methods being given as numbers even though they are really only identifiers. (If you do, your report ought to say what you did and why.)

Solution

The appropriate plot, with a categorical method and quantitative weight, is something like a boxplot. If you're not careful, `method` will get treated as a quantitative variable, which you don't want; the easiest way around that, for a boxplot at least, is to turn it into a factor like this:

```
ggplot(carbon, aes(x = factor(method), y = weight)) + geom_boxplot()
```

[8] It's probably better in a report to use language a bit more formal than *a bunch*. Something like *a number* would be better.

If you insist, you could do a faceted histogram (above and below, for preference):

```
ggplot(carbon, aes(x = weight)) + geom_histogram(bins = 5) +
  facet_wrap(~method, ncol = 1)
```

There are really not enough data values for a histogram to be of much help, so I don't like this as much.

If you are thinking ahead (we are going to be doing a t-test), then you'll realize that normality is the kind of thing we're looking for, in which case normal quantile plots would be the thing. However, we might have to be rather forgiving for method 2 since there are only 5 observations:

```
ggplot(carbon, aes(sample = weight)) +
  stat_qq() + stat_qq_line() +
  facet_wrap(~method)
```

I don't mind these coming out side by side, though I would rather have them squarer.

I would say, boxplots are the best, normal quantile plots are also acceptable, but expect to lose something for histograms because they offer only a rather crude comparison in this case.

■

(e) Comment briefly on what you see in your plot.

Solution

In boxplots, if that's what you drew, there are several things that deserve comment: the medians, the spreads and the shapes. The median for method 1 is a little bit lower than for method 2 (the means are probably more different, given the shapes of the boxes). The spread for method 2 is a lot bigger. (Looking forward, that suggests a Welch-Satterthwaite rather than a pooled test.) As for shape, the method 2 measurements seem more or less symmetric (the whiskers are equal anyway, even if the position of the median in the box isn't), but the method 1 measurements have a low outlier. The histograms are hard to compare. Try to say something about centre and spread and shape. I think the method 2 histogram has a slightly higher centre and definitely bigger spread. On my histogram for method 1, the distribution looks skewed left. If you did normal quantile plots, say something sensible about normality for each of the two methods. For method 1, I would say the low value is an outlier and the rest of the values look pretty straight. Up to you whether you think there is a curve on the plot (which would indicate skewness, but then that highest value is too high: it would be bunched up with the other values below 12.01 if there were really skewness). For method 2, it's really

hard to say anything since there are only five values. Given where the line goes, there isn't much you can say to doubt normality. Perhaps the best you can say here is that in a sample of size 5, it's difficult to assess normality at all.

■

(f) Carry out the most appropriate t-test. (You might like to begin another new section in your report here.)

Solution

This would be the Welch-Satterthwaite version of the two-sample t-test, since the two groups do appear to have different spreads:

```
t.test(weight ~ method, data = carbon)
```

```
    Welch Two Sample t-test

data:  weight by method
t = -1.817, df = 5.4808, p-value = 0.1238
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to
95 percent confidence interval:
 -0.027777288  0.004417288
sample estimates:
mean in group 1 mean in group 2
       12.00260        12.01428
```

Imagining that this is a report that would go to your boss, you ought to defend your choice of the Welch-Satterthwaite test (as I did above), and not just do the default t-test without comment.

If, in your discussion above, you thought the spreads were equal enough, then you should do the pooled t-test here, which goes like this:

```
t.test(weight ~ method, data = carbon, var.equal = T)
```

```
    Two Sample t-test

data:  weight by method
t = -2.1616, df = 13, p-value = 0.04989
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to
```

```
95 percent confidence interval:
 -2.335341e-02 -6.588810e-06
sample estimates:
mean in group 1 mean in group 2
       12.00260        12.01428
```

The point here is that you should do the right test based on your conclusion. Being consistent is the most important thing. (In this case, note that the P-values are very different. We'll get to that shortly.)

If we were doing this in SAS, as we see later, we'd get a test at the bottom of the output that compares the two variances. I feel that it's just as good to eyeball the spreads and make a call about whether they are "reasonably close". Or even, to always do the Welch-Satterthwaite test on the basis that it is pretty good even if the two populations have the same variance. (If this last point of view is one that you share, you ought to say something about that when you do your t-test.)

Extra: I guess this is a good place to say something about tests for comparing variances, given that you might be pondering that. There are several that I can think of, that R can do, of which I mention two.

The first is the F-test for variances that you might have learned in B57 (that is the basis for the ANOVA F-test):

```r
var.test(weight ~ method, data = carbon)
```

```
    F test to compare two variances

data:  weight by method
F = 0.35768, num df = 9, denom df = 4, p-value = 0.1845
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.04016811 1.68758230
sample estimates:
ratio of variances
         0.3576842
```

This, unfortunately, is rather dependent on the data in the two groups being approximately normal. Since we are talking variances rather than means, there is no Central Limit Theorem to rescue us for large samples (quite aside from the fact that these samples are *not* large). Since the ANOVA F-test is based on the same theory, this is why normality is also more important in ANOVA than it is in a t-test.

The second is Levene's test. This doesn't depend on normality (at least, not nearly so much), so I like it better in general:

```
library(car)
leveneTest(weight ~ factor(method), data = carbon)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  1  0.9887 0.3382
      13
```

Levene's test takes a different approach: first the absolute differences from the group medians are calculated, and then an ANOVA is run on the absolute differences. If, say, one of the groups has a larger spread than the other(s), its absolute differences from the median will tend to be bigger.[9] As for what we conclude here, well, neither of the variance tests show any significance at all, so from that point of view there is no evidence against using the pooled t-test. Having said that, the samples are small, and so it would be difficult to *prove* that the two methods have different variance, even if they actually did.[10]

Things are never as clear-cut as you would like. In the end, it all comes down to making a call and defending it.

■

(g) Do the most appropriate test you know that does not assume normally-distributed data.

Solution

That would be Mood's median test. Since I didn't say anything about building it yourself, feel free to use `smmr`:

```
library(smmr)
median_test(carbon, weight, method)
```

```
$table
     above
group above below
    1     3     6
```

[9]The use of absolute differences, and the median, downplays the influence of outliers. The assumption here is that the absolute differences from the medians are approximately normal, which seems a less big assumption than assuming the actual data are approximately normal.

[10]This is coming back to the *power* of something like Levene's test; the power of any test is not going to be very big if the sample sizes are small.

```
    2     4     1

$test
       what      value
1 statistic 2.80000000
2        df 1.00000000
3   P-value 0.09426431
```

As an aside, if you have run into a non-parametric test such as Mann-Whitney or Kruskal-Wallis that applies in this situation, be careful about using it here, because they have additional assumptions that you may not want to trust. Mann-Whitney started life as a test for "equal distributions".[11] This means that the null is equal location *and* equal spread, and if you reject the null, one of those has failed. But here, we suspect that equal spread will fail, so that the Mann-Whitney test may end up rejecting *whether or not* the medians are different, so it won't answer the question you want an answer to. Mood's median test doesn't have that problem; all it's saying if the null is true is that the medians are equal; the spreads could be anything at all.

The same kind of issues apply to the signed-rank test vs. the sign test. In the case of the signed-rank test, the extra assumption is of a symmetric distribution — to my mind, if you don't believe normality, you probably don't have much confidence in symmetry either. That's why I like the sign test and Mood's median test: in the situation where you don't want to be dealing with assumptions, these tests don't make you worry about that.

Another comment that you don't need to make is based on the not-quite-significance of the Mood test. The P-value is less than 0.10 but not less than 0.05, so it doesn't quite reach significance by the usual standard. But if you look up at the table, the frequencies seem rather unbalanced: 6 out of the remaining 9 weights in group 1 are below the overall median, but 4 out of 5 weights in group 2 are above. This seems as if it ought to be significant, but bear in mind that the sample sizes are small, and thus Mood's median test needs *very* unbalanced frequencies, which we don't quite have here.

■

(h) Discuss the results of your tests and what they say about the two methods for measuring the atomic weight of carbon. If it seems appropriate, put the discussion into a section called Conclusions.

Solution

Begin by pulling out the P-values for your preferred test(s) and say what they mean. The P-value for the Welch-Satterthwaite t-test is 0.1238, which indicates no difference in mean atomic weights between the two methods. The Mood median test gives a similarly non-significant

[11] The test goes back to the 1940s.

0.0943, indicating no difference in the *median* weights. If you think both tests are plausible, then give both P-values and do a compare-and-contrast with them; if you think that one of the tests is clearly preferable, then say so (and why) and focus on that test's results.

If you thought the pooled test was the right one, then you'll have a bit more discussion to do, since its P-value is 0.0499, and at $\alpha = 0.05$ this test disagrees with the others. If you are comparing this test with the Mood test, you ought to make some kind of reasoned recommendation about which test to believe.

As ever, be consistent in your reasoning.

Extra: this dataset, where I found it, was actually being used to illustrate a case where the pooled and the Welch-Satterthwaite tests disagreed. The authors of the original paper that used this dataset (a 1987 paper by Best and Rayner;[12] the data come from 1924!) point out that the pooled t-test can be especially misleading when the smaller sample is also the one with the larger variance. This is what happened here.

In the Best and Rayner paper, the Mood (or the Mann-Whitney) test was not being considered, but I think it's good practice to draw a picture and make a call about which test is appropriate.

■

11.14 Can caffeine improve your performance on a test?

Does caffeine help students do better on a certain test? To find out, 36 students were randomly allocated to three groups (12 in each group). Each student received a fixed number of cups of coffee while they were studying, but the students didn't know whether they were receiving all full-strength coffee ("high"), all decaf coffee ("low") or a 50-50 mixture of the two ("moderate"). For each subject, their group was recorded as well as their score on the test. The data are in link, as a `.csv` file.

(a) Read in and examine the data. How are the values laid out?

Solution

`read_csv` because it's a `.csv` file:

```
my_url <- "http://ritsokiguess.site/datafiles/caffeine.csv"
caffeine.untidy <- read_csv(my_url)
```

[12] Best, D. J., and J. C. W. Rayner. "Welch's Approximate Solution for the Behrens–Fisher Problem." Technometrics 29, no. 2 (May 1, 1987): 205–10. doi:10.1080/00401706.1987.10488211. The data set is near the end.

```
Rows: 12 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (4): Sub, High, Moderate, None

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
caffeine.untidy
```

```
# A tibble: 12 x 4
     Sub  High Moderate  None
   <dbl> <dbl>    <dbl> <dbl>
 1     1    72       68    68
 2     2    65       80    74
 3     3    68       64    59
 4     4    83       65    61
 5     5    79       69    65
 6     6    92       79    72
 7     7    69       80    80
 8     8    74       63    58
 9     9    78       69    65
10    10    83       70    60
11    11    88       83    78
12    12    71       75    75
```

The first column is the number of the subject (actually within each group, since each student only tried one amount of caffeine). Then follow the test scores for the students in each group, one group per column.

I gave the data frame a kind of dumb name, since (looking ahead) I could see that I would need a less-dumb name for the tidied-up data, and it seemed sensible to keep `caffeine` for that.

■

(b) Explain briefly how the data are not "tidy".

Solution

The last three columns are all scores on the test: that is, they all measure the same thing, so they should all be in the same column. Or, there should be a column of scores, and a separate

column naming the groups. Or, there were 36 observations in the data, so there should be 36 rows. You always have a variety of ways to answer these, any of which will do.

■

(c) Use a suitable tool from the `tidyverse` to create one column of test scores and and one column of group labels. Call your column of group labels `amount`. Is it a `factor`?

Solution

We are combining several columns into one, so this is `pivot_longer`:

```
caffeine.untidy %>%
  pivot_longer(-Sub, names_to = "amount", values_to = "score") -> caffeine
```

I didn't ask you to list the resulting data frame, but it is smart to at least look for yourself, to make sure `pivot_longer` has done what you expected.

```
caffeine
```

```
# A tibble: 36 x 3
     Sub amount   score
   <dbl> <chr>    <dbl>
 1     1 High        72
 2     1 Moderate    68
 3     1 None        68
 4     2 High        65
 5     2 Moderate    80
 6     2 None        74
 7     3 High        68
 8     3 Moderate    64
 9     3 None        59
10     4 High        83
# i 26 more rows
```

A column of amounts of caffeine, and a column of test scores. This is what we expected. There should be 12 each of the `amount`s, which you can check if you like:

```
caffeine %>% count(amount)
```

```
# A tibble: 3 x 2
  amount       n
  <chr>    <int>
```

```
1 High        12
2 Moderate    12
3 None        12
```

Indeed.

Note that `amount` is text, not a factor. Does this matter? We'll see.

This is entirely the kind of situation where you need `pivot_longer`, so get used to seeing where it will be useful.

■

(d) Obtain side-by-side boxplots of test scores by amount of caffeine.

Solution

```
ggplot(caffeine, aes(x = amount, y = score)) + geom_boxplot()
```

Note that this is *much more difficult* if you don't have a tidy data frame. (Try it and see.)

■

(e) Does caffeine amount seem to have an effect? If so, what kind of effect?

Solution

On average, exam scores seem to be higher when the amount of caffeine is higher (with the effect being particularly pronounced for High caffeine). If you want to, you can also say the the effect of caffeine seems to be small, relative to the amount of variability there is (there is a lot). The point is that you say *something* supported by the boxplot.

■

(f) Run a suitable analysis of variance to determine whether the mean test score is equal or unequal for the three groups. What do you conclude?

Solution

Something like this:

```
caff.1 <- aov(score ~ amount, data = caffeine)
summary(caff.1)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
amount       2  477.7  238.86   3.986 0.0281 *
Residuals   33 1977.5   59.92
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The P-value on the F-test is less than 0.05, so we reject the null hypothesis (which says that all the groups have equal means) in favour of the alternative: the group means are not all the same (one or more of them is different from the others).

Notice that the boxplot and the `aov` are quite happy for `amount` to be text rather than a factor (they actually do want a factor, but if the input is text, they'll create one).

■

(g) Why is it a good idea to run Tukey's method here?

Solution

The analysis of variance F-test is significant, so that the groups are not all the same. Tukey's method will tell us which group(s) differ(s) from the others. There are three groups, so there are differences to find that we don't know about yet.

■

(h) Run Tukey's method. What do you conclude?

Solution

This kind of thing:

```
caff.3 <- TukeyHSD(caff.1)
caff.3
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = score ~ amount, data = caffeine)

$amount
                    diff       lwr       upr     p adj
Moderate-High -4.750000 -12.50468  3.004679 0.3025693
None-High     -8.916667 -16.67135 -1.161987 0.0213422
None-Moderate -4.166667 -11.92135  3.588013 0.3952176
```

The high-caffeine group definitely has a higher mean test score than the no-caffeine group. (The Moderate group is not significantly different from either of the other groups.) Both the comparisons involving Moderate could go either way (the interval for the difference in means includes zero). The None-High comparison, however, is away from zero, so this is the significant one. As is usual, we are pretty sure that the difference in means (this way around) is negative, but we are not at all clear about how big it is, because the confidence interval is rather long.[13]

Extra: the normality and equal spreads assumptions look perfectly good, given the boxplots, and I don't think there's any reason to consider any other test. You might like to assess that with normal quantile plots:

```
ggplot(caffeine, aes(sample=score)) + stat_qq() +
  stat_qq_line() + facet_wrap(~amount, ncol=2)
```

[13]We'd need a lot more students to make it narrower, but this is not surprising since students vary in a lot of other ways that were not measured here.

There's nothing to worry about there normality-wise. If anything, there's a little evidence of *short* tails (in the None group especially), but you'll recall that short tails don't affect the mean and thus pose no problems for the ANOVA. Those three lines also have pretty much the same slope, indicating very similar spreads. Regular ANOVA is the best test here. (Running eg. Mood's median test would be a mistake here, because it doesn't use the data as efficiently (counting only aboves and belows) as the ANOVA does, and so the ANOVA will give a better picture of what differs from what.)

■

11.15 Reggae music

Reggae is a music genre that originated in Jamaica in the late 1960s. One of the most famous reggae bands was Bob Marley and the Wailers. In a survey, 729 students were asked to rate reggae music on a scale from 1, "don't like it at all" to 6, "like it a lot". We will treat the ratings as quantitative. Each student was also asked to classify their home town as one of "big city", "suburban", "small town", "rural". Does a student's opinion of reggae depend on the kind of home town they come from? The data are in http://ritsokiguess.site/datafiles/reggae.csv.

(a) Read in and display (some of) the data.

Solution

This is (evidently) a `.csv`, so:

```r
my_url <- "http://ritsokiguess.site/datafiles/reggae.csv"
reggae <- read_csv(my_url)
```

```
Rows: 729 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): home
dbl (1): rating

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
reggae
```

```
# A tibble: 729 x 2
   home     rating
   <chr>     <dbl>
 1 big city      1
 2 big city      2
 3 big city      2
 4 big city      2
 5 big city      2
 6 big city      2
 7 big city      2
 8 big city      2
 9 big city      3
10 big city      3
# i 719 more rows
```

The students shown are all from big cities, but there are others, as you can check by scrolling down.

■

(b) How many students are from each different size of town?

Solution

This is the usual kind of application of `count`:

```r
reggae %>% count(home)
```

```
# A tibble: 4 x 2
  home           n
  <chr>      <int>
1 big city      89
2 rural         96
3 small town   176
4 suburban     368
```

Another, equally good, way (you can ignore the warning):

```
reggae %>% group_by(home) %>%
summarize(n=n())
```

```
# A tibble: 4 x 2
  home           n
  <chr>      <int>
1 big city      89
2 rural         96
3 small town   176
4 suburban     368
```

Most of the students in this data set are from suburbia.

■

(c) Make a suitable graph of the two variables in this data frame.

Solution

One quantitative, one categorical: a boxplot, as ever:

```
ggplot(reggae, aes(x=home, y=rating)) + geom_boxplot()
```

Extra 1: the last three boxplots really are identical, because the medians, means, quartiles and extreme values are all equal. However, the *data values* are not all the same, as you see below.

Extra 2: I said that the ratings should be treated as quantitative, to guide you towards this plot. You could otherwise have taken the point of view that the ratings were (ordered) categorical, in which case the right graph would have been a grouped bar chart, as below. There is a question about which variable should be `x` and which should be `fill`. I am taking the point of view that we want to compare ratings within each category of `home`, which I think makes sense here (see discussion below), which breaks my "rule" that the categorical variable with fewer categories should be `x`.[14]

```
ggplot(reggae, aes(x=home, fill=factor(rating))) + geom_bar(position = "dodge")
```

[14]Perhaps a better word here would be *principle*, to convey the idea that you can do something else if it works better for your purposes.

■

(d) Discuss briefly why you might prefer to run Mood's median test to compare ratings among home towns.

Solution

The issue here is whether *all* of the rating distributions (within each category of `home`) are sufficiently close to normal in shape. The "big city" group is clearly skewed to the left. This is enough to make us favour Mood's median test over ANOVA.

A part-marks answer is to note that the big-city group has smaller spread than the other groups (as measured by the IQR). This is answering the wrong question, though. Remember the process: first we assess normality. If that fails, we use Mood's median test. Then, with normality OK, we assess equal spreads. If *that* fails, we use Welch ANOVA, and if both normality and equal spreads pass, we use regular ANOVA.

■

(e) Suppose that somebody wanted to run Welch ANOVA on these data. What would be a reasonable argument to support that?

Solution

The argument would have to be that normality is all right, given the sample sizes. We found earlier that there are between 89 and 368 students in each group. These are large samples, and might be enough to overcome the non-normality we see.

The only real concern I have is with the big city group. This is the least normal, and also the smallest sample. The other groups seem to have the kind of non-normality that will easily be taken care of by the sample sizes we have.

Extra: the issue is really about the sampling distribution of the mean within each group. Does that look normal enough? This could be assessed by looking at each group, one at a time, and taking bootstrap samples. Here's the big-city group:

```
reggae %>% filter(home=="big city") -> bigs
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(bigs$rating, replace = T))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 12)
```

Not too much wrong with that. This shows that the sample size is indeed big enough to cope with the skewness.

You can do any of the others the same way.

If you're feeling bold, you can get hold of all three bootstrapped sampling distributions at once, like this:

```r
reggae %>%
  nest_by(home) %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(data$rating, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 12) +
  facet_wrap(~home, scales = "free")
```

All of these distributions look very much normal, so there is no cause for concern anywhere.

This was rather a lot of code, so let me take you through it. The first thing is that we want to treat the different students' homes separately, so the first step is this:

```r
reggae %>%
  nest_by(home)
```

```
# A tibble: 4 x 2
# Rowwise:  home
  home                      data
  <chr>      <list<tibble[,1]>>
```

```
1 big city             [89 x 1]
2 rural                [96 x 1]
3 small town          [176 x 1]
4 suburban            [368 x 1]
```

This subdivides the students' reggae ratings according to where their home is. The things in `data` are data frames containing a column `rating` for in each case the students who had the `home` shown.

Normally, we would start by making a dataframe with a column called `sim` that labels the 1000 or so simulations. This time, we want *four* sets of simulations, one for each `home`, which we can set up this way:

```
reggae %>%
  nest_by(home) %>%
  mutate(sim = list(1:1000))
```

```
# A tibble: 4 x 3
# Rowwise:  home
  home                     data sim
  <chr>      <list<tibble[,1]>> <list>
1 big city             [89 x 1] <int [1,000]>
2 rural                [96 x 1] <int [1,000]>
3 small town          [176 x 1] <int [1,000]>
4 suburban            [368 x 1] <int [1,000]>
```

The definition of `sim` happens by group, or rowwise, by `home` (however you want to look at it). Next, we need to spread out those `sim` values so that we'll have one row per bootstrap sample:

```
reggae %>%
  nest_by(home) %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim)
```

```
# A tibble: 4,000 x 3
# Groups:   home [4]
   home                   data   sim
   <chr>    <list<tibble[,1]>> <int>
 1 big city           [89 x 1]     1
 2 big city           [89 x 1]     2
```

```
 3 big city           [89 x 1]     3
 4 big city           [89 x 1]     4
 5 big city           [89 x 1]     5
 6 big city           [89 x 1]     6
 7 big city           [89 x 1]     7
 8 big city           [89 x 1]     8
 9 big city           [89 x 1]     9
10 big city           [89 x 1]    10
# i 3,990 more rows
```

$4 \times 1000 = 4000$ rows. Note that the `data` column now contains multiple copies of all the ratings for the students with that `home`, which seems wasteful, but it makes our life easier because what we want is a bootstrap sample from the right set of students, namely the `rating` column from the dataframe `data` in each row. Thus, from here out, everything is the same as we have done before: work rowwise, get a bootstrap sample , find its mean, plot it. The one thing we need to be careful of is to make a *separate* histogram for each `home`, since each of the *four* distributions need to look normal. I used different scales for each one, since they are centred in different places; this has the side benefit of simplifying the choice of the number of bins. (See what happens if you omit the `scales = "free"`.)

In any case, all is absolutely fine. We'll see how this plays out below.

■

(f) Run Mood's median test and display the output.

Solution

Data frame, quantitative column, categorical column:

```r
median_test(reggae, rating, home)
```

```
$table
            above
group        above below
  big city      51    21
  rural         25    49
  small town    64    89
  suburban     120   187

$test
       what        value
1 statistic 2.733683e+01
2        df 3.000000e+00
```

```
3   P-value 5.003693e-06
```

■

(g) Explain briefly why running pairwise median tests is a good idea, run them, and display the results.

Solution

The Mood's median test is significant, with a P-value of 0.000005, so the median ratings are not all the same. We want to find out how they differ.

(The table of aboves and belows, and for that matter the boxplot earlier, suggest that big-city will be different from the rest, but it is not clear whether there will be any other significant differences.)

```r
pairwise_median_test(reggae, rating, home)
```

```
# A tibble: 6 x 4
  g1         g2            p_value adj_p_value
  <chr>      <chr>           <dbl>       <dbl>
1 big city   rural      0.00000746  0.0000448
2 big city   small town 0.0000491   0.000295
3 big city   suburban   0.00000110  0.00000663
4 rural      small town 0.788       1
5 rural      suburban   0.740       1
6 small town suburban   0.963       1
```

■

(h) Summarize, as concisely as possible, how the home towns differ in terms of their students' ratings of reggae music.

Solution

The students from big cities like reggae more than students from other places. The other kinds of hometown do not differ significantly.

Extra 1: Given the previous discussion, you might be wondering how Welch ANOVA (and maybe even regular ANOVA) compare. Let's find out:

```r
oneway.test(rating~home,data=reggae)
```

```
    One-way analysis of means (not assuming equal variances)

data:  rating and home
F = 16.518, num df = 3.00, denom df = 257.07, p-value = 7.606e-10
```

and

```
gamesHowellTest(rating~factor(home),data=reggae)
```

```
    Pairwise comparisons using Games-Howell test

data: rating by factor(home)

           big city rural small town
rural      1.1e-07  -     -
small town 2.9e-06  0.74  -
suburban   4.9e-09  0.91  0.94


P value adjustment method: none

alternative hypothesis: two.sided
```

The conclusions are identical with Mood's median test, and the P-values are not that different, either.

This makes me wonder how an ordinary ANOVA with Tukey would have come out:

```
reggae %>%
aov(rating~home, data=.) %>%
TukeyHSD()
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = rating ~ home, data = .)

$home
```

```
                          diff        lwr        upr     p adj
rural-big city      -1.20681180 -1.8311850 -0.5824386 0.0000048
small town-big city -1.00510725 -1.5570075 -0.4532070 0.0000194
suburban-big city   -1.09404006 -1.5952598 -0.5928203 0.0000002
small town-rural     0.20170455 -0.3366662  0.7400753 0.7695442
suburban-rural       0.11277174 -0.3735106  0.5990540 0.9329253
suburban-small town -0.08893281 -0.4778062  0.2999406 0.9354431
```

Again, almost identical.

Extra 2: some Bob Marley and the Wailers for you:

- from 1980
- from 1973

Reggae music at its finest.

■

11.16 Watching TV and education

The General Social Survey is a large survey of a large number of people. One of the questions on the survey is "how many hours of TV do you watch in a typical day?" Another is "what is your highest level of education attained", on this scale:

- **HSorLess**: completed no more than high h school
- **College**: completed some form of college, either a community college (like Centennial) or a four-year university (like UTSC)
- **Graduate**: completed a graduate degree such as an MSc.

Do people with more education tend to watch more TV? We will be exploring this. The data are in http://ritsokiguess.site/datafiles/gss_tv.csv.

(a) Read in and display (some of) the data.

Solution

Exactly the usual:

```
my_url <- "http://ritsokiguess.site/datafiles/gss_tv.csv"
gss <- read_csv(my_url)
```

```
Rows: 905 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): degree
dbl (1): tvhours

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
gss
```

```
# A tibble: 905 x 2
   degree   tvhours
   <chr>      <dbl>
 1 HSorLess       1
 2 HSorLess       1
 3 HSorLess       4
 4 HSorLess       3
 5 HSorLess       6
 6 College        1
 7 HSorLess       5
 8 College        1
 9 College        3
10 HSorLess       1
# i 895 more rows
```

■

(b) For each level of education, obtain the number of observations, the mean and the median of the number of hours of TV watched.

Solution

`group_by` and `summarize`, using `n()` to get the number of observations (rather than `count` because you want some numerical summaries as well):

```
gss %>% group_by(degree) %>%
summarise(n=n(), mean=mean(tvhours), med=median(tvhours))
```

```
# A tibble: 3 x 4
  degree       n  mean   med
```

```
  <chr>    <int> <dbl> <dbl>
1 College    195  2.27     2
2 Graduate    70  1.84     1
3 HSorLess   640  3.33     3
```

■

(c) What does your answer to the previous part tell you about the shapes of the distributions of the numbers of hours of TV watched? Explain briefly.

Solution

In each of the three groups, the mean is greater than the median, so I think the distributions are skewed to the right. Alternatively, you could say that you expect to see some outliers at the upper end.

■

(d) Obtain a suitable graph of your data frame.

Solution

One quantitative variable and one categorical one, so a boxplot. (I hope you are getting the hang of this by now.)

```
ggplot(gss, aes(x=degree, y=tvhours)) + geom_boxplot()
```

■

(e) Does your plot indicate that your guess about the distribution shape was correct? Explain briefly.

Solution

I guessed before that the distributions would be right-skewed, and they indeed are, with the long upper tails. Or, if you suspected upper outliers, they are here as well.

Say what you guessed before, and how your graph confirms it (or doesn't, if it doesn't.)

■

(f) Run a suitable test to compare the average number of hours of TV watched for people with each amount of education. ("Average" could be mean or median, whichever you think is appropriate.)

Solution

From the boxplot, the distributions are definitely not all normal; in fact, none of them are. So we should use Mood's median test, thus:

```
median_test(gss, tvhours, degree)
```

```
$table
          above
group      above below
  College     67    70
  Graduate    18    36
  HSorLess   355   126

$test
       what        value
1 statistic 5.608269e+01
2        df 2.000000e+00
3   P-value 6.634351e-13
```

■

(g) What do you conclude from your test, in the context of the data?

Solution

The P-value of 6.6×10^{-13} is extremely small, so we conclude that not all of the education groups watch the same median amount of TV. Or, there are differences in the median amount of TV watched among the three groups.

An answer of "the education groups are different" is *wrong*, because you don't know that they are *all* different. It might be that some of them are different and some of them are the same. The next part gets into that.

■

(h) Why might you now want to run some kind of follow-up test? Run the appropriate thing and explain briefly what you conclude from it, in the context of the data.

Solution

The overall Mood test is significant, so there are some differences between the education groups, but we don't know where they are. Pairwise median tests will reveal where any differences are:

```
pairwise_median_test(gss, tvhours, degree)
```

```
# A tibble: 3 x 4
  g1       g2        p_value   adj_p_value
  <chr>    <chr>       <dbl>         <dbl>
1 College  Graduate 5.12e- 2 0.154
2 College  HSorLess 8.06e-10 0.00000000242
3 Graduate HSorLess 3.06e- 7 0.000000919
```

The people whose education is high school or less are significantly different from the other two education levels. The boxplot reveals that this is because they watch *more* TV on average. The college and graduate groups are not significantly different (in median TV watching).

Extra 1:

You might have been surprised that the College and Graduate medians were not significantly different. After all, they look quite different on the boxplot. Indeed, the P-value for comparing just those two groups is 0.0512, only just over 0.05. But remember that we are doing three tests at once, so the Bonferroni adjustment is to multiply the P-values by 3, so this P-value is "really" some way from being significant. I thought I would investigate this in more detail:

```
gss %>% filter(degree != "HSorLess") %>%
median_test(tvhours, degree)
```

```
$table
          above
group      above below
  College     67    70
  Graduate    18    36

$test
       what     value
1 statistic 3.8027625
2        df 1.0000000
3   P-value 0.0511681
```

The College group are about 50-50 above and below the overall median, but the Graduate group are two-thirds below. This suggests that the Graduate group watches less TV, and with these sample sizes I would have expected a smaller P-value. But it didn't come out that way.

You might also be concerned that there are in total more values below the grand median (106) than above (only 85). This must mean that there are a lot of data values *equal* to the grand median:

```
gss %>% filter(degree != "HSorLess") -> gss1
gss1 %>% summarize(med=median(tvhours))
```

```
# A tibble: 1 x 1
    med
  <dbl>
1     2
```

and

```
gss1 %>% count(tvhours)
```

```
# A tibble: 10 x 2
   tvhours     n
     <dbl> <int>
 1       0    15
 2       1    91
 3       2    74
 4       3    51
 5       4    17
 6       5     5
```

```
 7        6      6
 8        7      2
 9        8      3
10       12      1
```

Everybody gave a whole number of hours, and there are not too many different ones; in addition, a lot of them are equal to the grand median of 2.

Extra 2:

Regular ANOVA and Welch ANOVA should be non-starters here because of the non-normality, but you might be curious about how they would perform:

```r
gss.1 <- aov(tvhours~degree, data=gss)
summary(gss.1)
```

```
             Df Sum Sq Mean Sq F value   Pr(>F)    
degree        2    267  133.30   25.18 2.27e-11 ***
Residuals   902   4774    5.29                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
TukeyHSD(gss.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = tvhours ~ degree, data = gss)

$degree
                       diff        lwr       upr     p adj
Graduate-College  -0.4238095 -1.1763372 0.3287181 0.3831942
HSorLess-College   1.0598958  0.6181202 1.5016715 0.0000001
HSorLess-Graduate  1.4837054  0.8037882 2.1636225 0.0000011
```

and

```r
oneway.test(tvhours~degree, data=gss)
```

```
    One-way analysis of means (not assuming equal variances)

data:  tvhours and degree
F = 37.899, num df = 2.00, denom df = 206.22, p-value = 9.608e-15
```

```r
gamesHowellTest(tvhours~factor(degree), data=gss)
```

```
    Pairwise comparisons using Games-Howell test

data: tvhours by factor(degree)

         College Graduate
Graduate 0.12    -
HSorLess 2.4e-10 1.7e-10


P value adjustment method: none

alternative hypothesis: two.sided
```

The conclusions are actually identical to our Mood test, and the P-values are actually not all that much different. Which makes me wonder just how bad the sampling distributions of the sample means are. Bootstrap to the rescue:

```r
gss %>%
  nest_by(degree) %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(data$tvhours, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 12) +
  facet_wrap(~degree, scales = "free")
```

Coding this made my head hurt, but building it one line at a time, I pretty much got it right first time. In words:

- "compress" the dataframe to get one row per degree and a list-column called `data` with the number of hours of TV watched for each person with that `degree`
- generate 1000 `sim`s for each `degree` (to guide the taking of bootstrap samples shortly)
- organize into one row per `sim`
- then take bootstrap samples as normal and work out the mean of each one
- make histograms for each `degree`, using a different scale for each one. (This has the advantage that the normal number of `bins` will work for all the histograms.)

If you are not sure about what happened, run it one line at a time and see what the results look like after each one.

Anyway, even though the data was very much not normal, these sampling distributions are very normal-looking, suggesting that something like Welch ANOVA would have been not nearly as bad as you would have guessed. This is evidently because of the big sample sizes. (This also explains why the two other flavours of ANOVA gave results very similar to Mood's median test.)

■

11.17 Death of poets

Some people believe that poets, especially female poets, die younger than other types of writer. William Butler Yeats[15] wrote:

> She is the Gaelic[16] muse, for she gives inspiration to those she persecutes. The Gaelic poets die young, for she is restless, and will not let them remain long on earth.

A literature student wanted to investigate this, and so collected a sample of 123 female writers (of three different types), and noted the age at death of each writer.

The data are in http://ritsokiguess.site/datafiles/writers.csv.

(a) Read in and display (some of) the data.

Solution

The usual:

```
my_url <- "http://ritsokiguess.site/datafiles/writers.csv"
writers <- read_csv(my_url)
```

```
Rows: 123 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Type
dbl (2): Type1, Age

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
writers
```

```
# A tibble: 123 x 3
   Type1 Type     Age
   <dbl> <chr>  <dbl>
 1     1 Novels    57
 2     1 Novels    90
 3     1 Novels    67
```

[15] An Irish, that is to say, Gaelic, poet (see below), but a male one.

[16] Gaelic is a language of Scotland and Ireland, and the culture of the people who speak it.

```
 4     1 Novels    56
 5     1 Novels    90
 6     1 Novels    72
 7     1 Novels    56
 8     1 Novels    90
 9     1 Novels    80
10     1 Novels    74
# i 113 more rows
```

There are indeed 123 writers. The second column shows the principal type of writing each writer did, and the third column shows their age at death. The first column is a numerical code for the type of writing, which we ignore (since we can handle the text writing type).

■

(b) Make a suitable plot of the ages and types of writing.

Solution

As usual, one quantitative and one categorical, so a boxplot:

```
ggplot(writers, aes(x=Type, y=Age)) + geom_boxplot()
```

At this point, a boxplot is best, since right now you are mostly after a general sense of what is going on, rather than assessing normality in particular (that will come later).

■

(c) Obtain a summary table showing, for each type of writing, the number of writers of that type, along with the mean, median and standard deviation of their ages at death.

Solution

The customary `group_by` and `summarize`:

```
writers %>% group_by(Type) %>%
summarize(n=n(), mean=mean(Age), med=median(Age), sd=sd(Age))
```

```
# A tibble: 3 x 5
  Type           n  mean   med    sd
  <chr>      <int> <dbl> <dbl> <dbl>
1 Nonfiction    24  76.9  77.5  14.1
2 Novels        67  71.4  73    13.1
3 Poems         32  63.2  68    17.3
```

■

(d) Run a complete analysis, starting with an ordinary (not Welch) analysis of variance, that ends with a conclusion in the context of the data and an assessment of assumptions.

Solution

I've left this fairly open-ended, to see how well you know what needs to be included and what it means. There is a lot of room here for explanatory text to show that you know what you are doing. One output followed by another *without* any explanatory text suggests that you are just copying what I did without any idea about why you are doing it.

The place to start is the ordinary (not Welch) ANOVA. You may not think that this is the best thing to do (you'll have a chance to talk about that later), but I wanted to make sure that you practiced the procedure:

```
writers.1 <- aov(Age~Type, data=writers)
summary(writers.1)
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)
Type         2   2744  1372.1   6.563 0.00197 **
Residuals  120  25088   209.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This says that the mean ages at death of the three groups of writers are not all the same, or that there are differences among those writers (in terms of mean age at death). "The mean ages of the types of writer are different" is not accurate enough, because it comes too close to saying that *all three* groups are different, which is more than you can say right now.

The F-test is significant, meaning that there are some differences among[17] the means, and Tukey's method will enable us to see which ones differ:

```r
TukeyHSD(writers.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Age ~ Type, data = writers)

$Type
                       diff       lwr       upr     p adj
Novels-Nonfiction  -5.427239 -13.59016  2.735681 0.2591656
Poems-Nonfiction  -13.687500 -22.95326 -4.421736 0.0018438
Poems-Novels       -8.260261 -15.63375 -0.886772 0.0240459
```

There is a significant difference in mean age at death between the poets and both the other types of writer. The novelists and the nonfiction writers do not differ significantly in mean age at death.

We know from the boxplots (or the summary table) that this significant difference was because the poets died *younger* on average, which is exactly what the literature student was trying to find out. Thus, female poets really do die younger on average than female writers of other types. It is best to bring this point out, since this is the reason we (or the literature student) were doing this analysis in the first place. See Extra 1 for more.

So now we need to assess the assumptions on which the ANOVA depends.

The assumption we made is that the ages at death of the authors of each different type had approximately a normal distribution (given the sample sizes) with approximately equal spread. The boxplots definitely look skewed to the left (well, not the poets so much, but the others

[17] There might be differences between two things, but among three or more.

definitely). So now consider the sample sizes: 24, 67, and 32 for the three groups (respectively), and make a call about whether you think the normality is good enough. You are certainly entitled to declare the two outliers on the nonfiction writers to be too extreme given a sample size of only 24. Recall that once one sample fails normality, that's all you need.

Now, since you specifically want normality, you could reasonably look at normal quantile plots instead of the boxplots. Don't just get normal quantile plots, though; say something about why you want them instead of the boxplots you drew earlier:

```
ggplot(writers, aes(sample = Age)) +
stat_qq() + stat_qq_line() +
facet_wrap(~Type)
```

I see that the Nonfiction writers have two outliers at the low end (and are otherwise not bad); the writers of Novels don't go up high enough (it's almost as if there is some magic that stops them living beyond 90!); the writers of Poems have a short-tailed distribution. You'll remember that short tails are not a problem, since the mean is still descriptive of such a distribution; it's *long* tails or outliers or skewness that you need to be worried about. The outliers in the Nonfiction writers are the biggest concern.

Are you concerned that these outliers are a problem, given the sample size? There are only 24 nonfiction writers (from your table of means earlier), so the Central Limit Theorem will help a bit. Make a call about whether these outliers are a big enough problem. You can go either way on this, as long as you raise the relevant issues.

Another approach you might take is to look at the P-values. The one in the F-test is really small, and so is one of the ones in the Tukey. So even if you think the analysis is a bit off, those conclusions are not likely to change. The 0.02 P-value in the Tukey, however, is another story. This could become non-significant in actual fact if the P-value is not to be trusted.

Yet another approach (looking at the bootstrapped sampling distributions of the sample means) is in Extra 3. This gets more than a little hairy with three groups, especially doing it the way I do.

If you think that the normality is not good enough, it's a good idea to suggest that we might do a Mood's Median Test instead, and you could even do it (followed up with pairwise median tests). If you think that normality is all right, you might then look at the spreads. I think you ought to conclude that these are close enough to equal (the SDs from the summary table or the heights of the boxes on the boxplots), and so there is no need to do a Welch ANOVA. (Disagree if you like, but be prepared to make the case.)

I have several Extras:

Extra 1: having come to that tidy conclusion, we really ought to back off a bit. These writers were (we assume) a random sample of some population, but they were actually mostly Americans, with a few Canadian and Mexican writers. So this appears to be true at least for North American writers. But this is (or might be) a different thing to the Yeats quote about female Gaelic poets.

There is a more prosaic reason. It is harder (in most places, but especially North America) to get poetry published than it is to find a market for other types of writing. (A would-be novelist, say, can be a journalist or write for magazines to pay the bills while they try to find a publisher for their novel.) Thus a poet is living a more precarious existence, and that might bring about health problems.

Extra 2: with the non-normality in mind, maybe Mood's median test is the thing:

```
median_test(writers, Age, Type)
```

```
$table
            above
group        above below
  Nonfiction    17     6
  Novels        33    30
  Poems         10    22

$test
       what       value
1 statistic 9.872664561
2        df 2.000000000
```

```
3   P-value 0.007180888
```

The P-value here is a bit bigger than for the F-test, but it is still clearly significant. Hence, we do the pairwise median tests to find out which medians differ:

```
pairwise_median_test(writers, Age, Type)
```

```
# A tibble: 3 x 4
  g1         g2     p_value adj_p_value
  <chr>      <chr>    <dbl>       <dbl>
1 Nonfiction Novels 0.0531      0.159
2 Nonfiction Poems  0.00119     0.00358
3 Novels     Poems  0.0142      0.0426
```

The conclusion here is exactly the same as for the ANOVA. The P-values have moved around a bit, though: the first one is a little closer to significance (remember, look at the last column since we are doing three tests at once) and the last one is now only just significant.

```
writers %>% group_by(Type) %>%
summarize(n=n(), mean=mean(Age), med=median(Age), sd=sd(Age))
```

```
# A tibble: 3 x 5
  Type           n  mean   med    sd
  <chr>      <int> <dbl> <dbl> <dbl>
1 Nonfiction    24  76.9  77.5  14.1
2 Novels        67  71.4  73    13.1
3 Poems         32  63.2  68    17.3
```

In both of these two cases (Nonfiction-Novels and Novels-Poems), the medians are closer together than the means are. That would explain why the Novels-Poems P-value would increase, but not why the Nonfiction-Novels one would decrease.

I would have no objection *in general* to your running a Mood's Median Test on these data, but the point of *this* problem was to give you practice with `aov`.

Extra 3: the other way to assess if the normality is OK given the sample sizes is to obtain bootstrap sampling distributions of the sample means for each `Type`. The sample size for the novelists is 67, so I would expect the skewness there to be fine, but the two outliers among the Nonfiction writers may be cause for concern, since there are only 24 of those altogether.

Let's see if we can do all three at once (I like living on the edge). I take things one step at a time, building up a pipeline as I go. Here's how it starts:

```r
writers %>% nest_by(Type)
```

```
# A tibble: 3 x 2
# Rowwise:  Type
  Type                      data
  <chr>      <list<tibble[,2]>>
1 Nonfiction           [24 x 2]
2 Novels               [67 x 2]
3 Poems                [32 x 2]
```

The thing `data` is a so-called list-column. The dataframes we have mostly seen so far are like spreadsheets, in that each "cell" or "entry" in a dataframe has something like a number or a piece of text in it (or, occasionally, a thing that is True or False, or a date). Tibble-type dataframes are more flexible than that, however: each cell of a dataframe could contain *anything*.

In this one, the three things in the column `data` are each *dataframes*,[18] containing the column called `Age` from the original dataframe. These are the ages at death of the writers of that particular `Type`. These are the things we want bootstrap samples of.

I'm not at all sure how this is going to go, so let's shoot for just 5 bootstrap samples to start with. If we can get it working, we can scale up the number of samples later, but having a smaller number of samples is easier to look at:

```r
writers %>% nest_by(Type) %>%
  mutate(sim = list(1:5))
```

```
# A tibble: 3 x 3
# Rowwise:  Type
  Type                      data sim
  <chr>      <list<tibble[,2]>> <list>
1 Nonfiction           [24 x 2] <int [5]>
2 Novels               [67 x 2] <int [5]>
3 Poems                [32 x 2] <int [5]>
```

Let me break off at this point to say that we want 1000 bootstrap samples for the writers of each type, so this is the kind of thing we need to start with. `nest_by` has an implied `rowwise`, so we get three lots of values in `sim`; the `list` is needed since each one is five values rather than just one. The next stage is to unnest these, and then do *another* `rowwise` to work with

[18] Like those Russian dolls.

all the (more) rows of the dataframe we now have. After that, the process should look more or less familiar:

```
writers %>% nest_by(Type) %>%
  mutate(sim = list(1:5)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(data$Age, replace = TRUE)))
```

```
# A tibble: 15 x 4
# Rowwise:  Type
   Type                      data   sim my_sample
   <chr>      <list<tibble[,2]>> <int> <list>
 1 Nonfiction           [24 x 2]     1 <dbl [24]>
 2 Nonfiction           [24 x 2]     2 <dbl [24]>
 3 Nonfiction           [24 x 2]     3 <dbl [24]>
 4 Nonfiction           [24 x 2]     4 <dbl [24]>
 5 Nonfiction           [24 x 2]     5 <dbl [24]>
 6 Novels               [67 x 2]     1 <dbl [67]>
 7 Novels               [67 x 2]     2 <dbl [67]>
 8 Novels               [67 x 2]     3 <dbl [67]>
 9 Novels               [67 x 2]     4 <dbl [67]>
10 Novels               [67 x 2]     5 <dbl [67]>
11 Poems                [32 x 2]     1 <dbl [32]>
12 Poems                [32 x 2]     2 <dbl [32]>
13 Poems                [32 x 2]     3 <dbl [32]>
14 Poems                [32 x 2]     4 <dbl [32]>
15 Poems                [32 x 2]     5 <dbl [32]>
```

That seems to be about the right thing; the bootstrap samples appear to be the right size, considering how many writers of each type our dataset had. From here, work out the mean of each sample:

```
writers %>% nest_by(Type) %>%
  mutate(sim = list(1:5)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(data$Age, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample))
```

```
# A tibble: 15 x 5
```

```
# Rowwise:  Type
   Type                      data   sim my_sample  my_mean
   <chr>      <list<tibble[,2]>> <int> <list>       <dbl>
 1 Nonfiction           [24 x 2]     1 <dbl [24]>    73.0
 2 Nonfiction           [24 x 2]     2 <dbl [24]>    74.8
 3 Nonfiction           [24 x 2]     3 <dbl [24]>    79
 4 Nonfiction           [24 x 2]     4 <dbl [24]>    76.6
 5 Nonfiction           [24 x 2]     5 <dbl [24]>    77.6
 6 Novels               [67 x 2]     1 <dbl [67]>    71.1
 7 Novels               [67 x 2]     2 <dbl [67]>    73.1
 8 Novels               [67 x 2]     3 <dbl [67]>    73.5
 9 Novels               [67 x 2]     4 <dbl [67]>    71.5
10 Novels               [67 x 2]     5 <dbl [67]>    70.3
11 Poems                [32 x 2]     1 <dbl [32]>    65.2
12 Poems                [32 x 2]     2 <dbl [32]>    65.7
13 Poems                [32 x 2]     3 <dbl [32]>    64.8
14 Poems                [32 x 2]     4 <dbl [32]>    58.7
15 Poems                [32 x 2]     5 <dbl [32]>    57.9
```

and then you could plot those means. This seems to be working, so let's scale up to 1000 simulations, and make normal quantile plots of the bootstrapped sampling distributions, one for each Type of writer:

```r
writers %>% nest_by(Type) %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(data$Age, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() +
  stat_qq_line() + facet_wrap(~Type, scales = "free")
```

These three normal quantile plots are all acceptable, to my mind, although the Nonfiction one, with the two outliers and the smallest sample size, is still a tiny bit skewed to the left. Apart from that, the three sampling distributions of the sample means are close to normal, so our `aov` is much better than you might have thought from looking at the boxplots. That's the result of having large enough samples to get help from the Central Limit Theorem.

■

11.18 Religion and studying

Many students at a certain university were asked about the importance of religion in their lives (categorized as "not", "fairly", or "very" important), and also about the number of hours they spent studying per week. (This was part of a much larger survey.) We want to see whether there is any kind of relationship between these two variables. The data are in here.

(a) Read in and display (some of) the data.

Solution

The usual. This is a straightforward one:

```
my_url <- "http://ritsokiguess.site/datafiles/student_relig.csv"
student <- read_csv(my_url)
```

```
Rows: 686 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): ReligImp
dbl (1): StudyHrs

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
student
```

```
# A tibble: 686 x 2
   ReligImp StudyHrs
   <chr>       <dbl>
 1 Fairly          3
 2 Fairly         30
 3 Fairly         16
 4 Not             4
 5 Not            12
 6 Fairly         20
 7 Fairly          4
 8 Not            15
 9 Fairly          7
10 Fairly         40
# i 676 more rows
```

686 students, with columns obviously named for religious importance and study hours.

Extra:

I said this came from a bigger survey, actually this one:

```r
my_url <- "http://ritsokiguess.site/datafiles/student0405.csv"
student0 <- read_csv(my_url)
```

```
Rows: 690 Columns: 7
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): Sex, ReligImp, Seat
dbl (4): GPA, MissClass, PartyDays, StudyHrs
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
student0
```

```
# A tibble: 690 x 7
   Sex      GPA ReligImp MissClass Seat   PartyDays StudyHrs
   <chr>  <dbl> <chr>        <dbl> <chr>      <dbl>    <dbl>
 1 Female  3.7  Fairly           1 Back           5        3
 2 Male    3.2  Fairly           3 Front          3       30
 3 Female  3.01 Fairly           0 Middle         8       16
 4 Female  3.77 Not              0 Middle         0        4
 5 Male    3.28 Not              0 Middle         8       12
 6 Female  2.8  Fairly           0 Middle         2       20
 7 Male    2.5  Fairly           3 Back           1        4
 8 Male    3.11 Not              0 Front          2       15
 9 Male    3.15 Fairly           2 Back          15        7
10 Male    3.44 Fairly           0 Middle         1       40
# i 680 more rows
```

There are four extra rows here. Why? Let's look at a `summary` of the dataframe:

```r
summary(student0)
```

```
     Sex                 GPA          ReligImp           MissClass
 Length:690         Min.   :1.500   Length:690         Min.   :0.0000
 Class :character   1st Qu.:2.930   Class :character   1st Qu.:0.0000
 Mode  :character   Median :3.200   Mode  :character   Median :1.0000
                    Mean   :3.179                      Mean   :0.9064
                    3rd Qu.:3.515                      3rd Qu.:1.0000
                    Max.   :4.000                      Max.   :6.0000
                    NA's   :3                          NA's   :1
     Seat             PartyDays         StudyHrs
 Length:690         Min.   : 0.000   Min.   : 0.00
 Class :character   1st Qu.: 3.000   1st Qu.: 6.25
 Mode  :character   Median : 7.000   Median :10.00
                    Mean   : 7.501   Mean   :13.16
                    3rd Qu.:11.000   3rd Qu.:16.00
                    Max.   :31.000   Max.   :70.00
                                     NA's   :4
```

You get information about each variable. For the text variables, you don't learn much, only how many there are. (See later for more on this.) For each of the four quantitative variables, you see some stats about each one, along with a count of missing values. The study hours variable is evidently skewed to the right (mean bigger than median), which we will have to think about later.

R also has a "factor" variable type, which is the "official" way to handle categorical variables in R. Sometimes it matters, but most of the time leaving categorical variables as text is just fine. `summary` handles these differently. My second line of code below says "for each variable that is text, make it into a factor":

```
student0 %>%
mutate(across(where(is.character), ~factor(.))) %>%
summary()
```

```
     Sex           GPA          ReligImp     MissClass          Seat
 Female:382   Min.   :1.500   Fairly:319   Min.   :0.0000   Back  :134
 Male  :308   1st Qu.:2.930   Not   :222   1st Qu.:0.0000   Front :151
              Median :3.200   Very  :149   Median :1.0000   Middle:404
              Mean   :3.179                Mean   :0.9064   NA's  :  1
              3rd Qu.:3.515                3rd Qu.:1.0000
              Max.   :4.000                Max.   :6.0000
              NA's   :3                    NA's   :1
   PartyDays         StudyHrs
 Min.   : 0.000   Min.   : 0.00
 1st Qu.: 3.000   1st Qu.: 6.25
 Median : 7.000   Median :10.00
 Mean   : 7.501   Mean   :13.16
 3rd Qu.:11.000   3rd Qu.:16.00
 Max.   :31.000   Max.   :70.00
                  NA's   :4
```

For factors, you also get how many observations there are in each category, and the number of missing values, which we didn't get before. However, `ReligImp` does not have any missing values.

I said there were four missing values for study hours, that is, four students who left that blank on their survey. We want to get rid of those students (that is, remove those whole rows), and, to simplify things for you, let's keep only the study hours and importance of religion columns. That goes like this:

```r
student0 %>% drop_na(StudyHrs) %>%
select(ReligImp, StudyHrs)
```

```
# A tibble: 686 x 2
   ReligImp StudyHrs
   <chr>       <dbl>
 1 Fairly          3
 2 Fairly         30
 3 Fairly         16
 4 Not             4
 5 Not            12
 6 Fairly         20
 7 Fairly          4
 8 Not            15
 9 Fairly          7
10 Fairly         40
# i 676 more rows
```

Then I saved that for you. 686 rows instead of 690, having removed the four rows with missing `StudyHrs`.

Another (better, but more complicated) option is to use the package `pointblank`, which produces much more detailed data validation reports. You would start that by piping your data into `scan_data()` to get a (very) detailed report of missingness and data values, and then you can check your data for particular problems, such as missing values, or values bigger or smaller than they should be, for the variables you care about. See here for more.

■

(b) Obtain the number of observations and the mean and standard deviation of study hours for each level of importance.

Solution

`group_by` and `summarize` (spelling the latter with s or z as you prefer):

```r
student %>% group_by(ReligImp) %>%
summarize(n=n(), mean_sh=mean(StudyHrs), sd_sh=sd(StudyHrs))
```

```
# A tibble: 3 x 4
  ReligImp     n mean_sh sd_sh
  <chr>    <int>   <dbl> <dbl>
```

```
1 Fairly     316    12.9  9.00
2 Not        222    11.7  8.49
3 Very       148    16.0 11.3
```

■

(c) Comment briefly on how the groups compare in terms of study hours.

Solution

The students who think religion is very important have a higher mean number of study hours. The other two groups seem similar.

As far as the SDs are concerned, make a call. You could say that the very-important group also has a (slightly) larger SD, or you could say that the SDs are all very similar.
I would actually favour the second one, but this is going to be a question about Welch ANOVA, so go whichever way you like.

■

(d) Make a suitable graph of this data set.

Solution

This kind of data is one quantitative and one categorical variable, so once again a boxplot:

```
ggplot(student, aes(x=ReligImp, y=StudyHrs)) + geom_boxplot()
```

■

(e) The statistician in this study decided that the data were sufficiently normal in shape given the (very large) sample sizes, but was concerned about unequal spreads among the three groups. Given this, run a suitable analysis and display the output. (This includes a suitable follow-up test, if warranted.)

Solution

Normal-enough data (in the statistician's estimation) and unequal spreads means a Welch ANOVA:

```
oneway.test(StudyHrs~ReligImp, data=student)
```

```
    One-way analysis of means (not assuming equal variances)

data:  StudyHrs and ReligImp
F = 7.9259, num df = 2.0, denom df = 350.4, p-value = 0.0004299
```

```
gamesHowellTest(StudyHrs~factor(ReligImp), data=student)
```

```
    Pairwise comparisons using Games-Howell test

data: StudyHrs by factor(ReligImp)

     Fairly  Not
Not  0.26035 -
Very 0.00906 0.00026


P value adjustment method: none

alternative hypothesis: two.sided
```

Games-Howell is the suitable follow-up here, to go with the Welch ANOVA. It is warranted because the Welch ANOVA was significant.

Make sure you have installed and loaded `PMCMRplus` before trying the second half of this.

Extra: for large data sets, boxplots make it look as if the outlier problem is bad, because a boxplot of a large amount of data will almost certainly contain some outliers (according to Tukey's definition). Tukey envisaged a boxplot as something you could draw by hand for a smallish data set, and couldn't foresee something like R and the kind of data we might be able to deal with. To show you the kind of thing I mean, let's draw some random samples of varying sizes from normal distributions, which should not have outliers, and see how their boxplots look:

```
tibble(n=c(10, 30, 100, 300)) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(n))) %>%
  unnest(my_sample) %>%
  ggplot(aes(x = factor(n), y = my_sample)) + geom_boxplot()
```

As the sample size gets bigger, the number of outliers gets bigger, and the whiskers get longer. All this means is that in a larger sample, you are more likely to see a small number of values that are further out, and that is not necessarily a reason for concern. Here, the outliers are only one value out of 100 and two out of 300, but they have what looks like an outsize influence

on the plot. In the boxplot for our data, the distributions were a bit skewed, but the outliers may not have been as much of a problem as they looked.

■

(f) What do you conclude from your analysis of the previous part, in the context of the data?

Solution

The Welch ANOVA was significant, so the religious-importance groups are not all the same in terms of mean study hours, and we need to figure out which groups differ from which. (Or say this in the previous part if you wish.)

The students for whom religion was very important had a significantly different mean number of study hours than the other students; the Fairly and Not groups were not significantly different from each other. Looking back at the means (or the boxplots), the significance was because the Very group studied for *more* hours than the other groups. It seems that religion has to be very important to a student to positively affect how much they study.

Extra: you might have been concerned that the study hours within the groups were not nearly normal enough to trust the Welch ANOVA. But the groups were large, so there is a lot of help from the Central Limit Theorem. Enough? Well, that is hard to judge.

My take on this is to bootstrap the sampling distribution of the sample mean for each group. If *that* looks normal, then we ought to be able to trust the F-test (regular or Welch, as appropriate). The code is complicated (I'll explain the ideas below):

```
student %>%
  nest_by(ReligImp) %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(data$StudyHrs, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line() +
  facet_wrap(~ReligImp, scales = "free")
```

To truly understand what's going on, you probably need to run this code one line at a time.

Anyway, these normal quantile plots are *very* normal. This says that the sampling distributions of the sample means are *very much* normal in shape, which means that the sample sizes are definitely large enough to overcome the apparently bad skewness that we saw on the boxplots. In other words, using a regular or Welch ANOVA will be perfectly good; there is no need to reach for Mood's median test here, despite what you might think from looking at the boxplots, because the sample sizes are so large.

The code, line by line:

- create mini-data-frames called `data`, containing one column called `StudyHrs`, for each `ReligImp` group
- set up for 1000 bootstrap samples for each group, and (next line) arrange for one row per bootstrap sample
- work rowwise
- generate the bootstrap samples
- work out the mean of each bootstrap sample
- plot normal quantile plots of them, using different facets for each group.

Finally, you might have wondered whether we needed to do Welch:

```
student.1 <- aov(StudyHrs~ReligImp, data=student)
summary(student.1)
```

```
             Df Sum Sq Mean Sq F value   Pr(>F)
ReligImp      2   1721   860.7   9.768 6.57e-05 ***
Residuals   683  60184    88.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
TukeyHSD(student.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = StudyHrs ~ ReligImp, data = student)

$ReligImp
                 diff        lwr       upr     p adj
Not-Fairly  -1.195917 -3.1267811 0.7349462 0.3135501
Very-Fairly  3.143047  0.9468809 5.3392122 0.0023566
Very-Not     4.338964  1.9991894 6.6787385 0.0000454
```

It didn't make much difference, and the conclusions are identical. So I think either way would have been defensible.

The value of doing Tukey is that we get confidence intervals for the difference of means between each group, and this gives us an "effect size": the students for whom religion was very important studied on average three or four hours per week more than the other students, and you can look at the confidence intervals to see how much uncertainty there is in those estimates. Students vary a lot in how much they study, but the sample sizes are large, so the intervals are not that long.

■

12 Writing reports

```
library(tidyverse)
```

12.1 Atomic weight of carbon

The atomic weight of the chemical element carbon is 12. Two methods of measuring the atomic weight of samples of carbon were compared. The results are shown in link. The methods are labelled 1 and 2. The first task is to find out whether the two methods have different "typical" measures (mean or median, as appropriate) of the atomic weight of carbon.

For this question, compose a report in R Markdown. (R Markdown is what you use in an R Notebook, but you can also have a separate R Markdown document from which you can produce HTML, Word etc. output.) See part (a) for how to get this started.

Your report should read like an actual report, not just the answers to some questions that I set you. To help with that, write some text that links the parts of the report together smoothly, so that it reads as a coherent whole. The grader had 3 discretionary marks to award for the overall quality of your writing. The scale for this was:

- 3 points: excellent writing. The report flows smoothly, is easy to read, and contains everything it should (and nothing it shouldn't).
- 2 points: satisfactory writing. Not the easiest to read, but says what it should, and it looks at least somewhat like a report rather than a string of answers to questions.
- 1 point: writing that is hard to read or to understand. If you get this (or 0), you should consider what you need to do to improve when you write your project.
- 0 points: you answered the questions, but you did almost nothing to make it read like a report.

(a) Create a new R Markdown document. To do this, in R Studio, select File, New File, R Markdown. Type the report title and your name in the boxes, and leave the output on the default HTML. Click OK.

(b) Write an introduction that explains the purpose of this study and the data collected in your own words.

(c) Begin an appropriately-titled new section in your report, read the data into R and display the results.

(d) Make an appropriate plot to compare the measurements obtained by the two methods. You might need to do something about the two methods being given as numbers even though they are really only identifiers. (If you do, your report ought to say what you did and why.)

(e) Comment briefly on what you see in your plot.

(f) Carry out the most appropriate t-test. (You might like to begin another new section in your report here.)

(g) Do the most appropriate test you know that does not assume normally-distributed data.

(h) Discuss the results of your tests and what they say about the two methods for measuring the atomic weight of carbon. If it seems appropriate, put the discussion into a section called Conclusions.

12.2 Sparrowhawks

(This is a report-writing question, but it also uses some regression techniques from later in the course.)

One of nature's patterns is the relationship between the percentage of adult birds in a colony that return from the previous year, and the number of new adults that join the colony. Data for 13 colonies of sparrowhawks can be found at link. The columns are the percentage of adults returning from the previous year, and the number of new adults that join the colony.

(a) Create a new R Markdown report, give it a suitable title, and ask for HTML output. Answer the questions that follow in your report. At any stage, you can Knit HTML to see how the report looks so far.

(b) Read in the data and display the first few values. Add some text saying how many rows of data there are.

(c) Create a new section entitled "Exploratory analysis", and create a scatterplot for predicting number of new adults from the percentage of returning adults. Describe what you see, adding some suitable text to your report.

(d) Obtain the correlation between the two variables. Is this consistent with the scatterplot? Explain briefly. (The R function you need is `cor`. You can feed it a data frame.)

(e) Obtain the regression line for predicting the number of new adults from the percentage of returning adults.

(f) What are the intercept and slope of your regression line? Is the slope significant? What does that mean, in the context of the data?

(g) Create a scatterplot of the data with the regression line on it.

(h) For short-lived birds, the association between these two variables is positive: changes in weather and food supply cause the populations of new and returning birds to increase together. For long-lived territorial birds, however, the association is negative because returning birds claim their territories in the colony and do not leave room for new recruits. Which type of species is the sparrowhawk? Add a short Conclusions section to your report with discussion of this issue.

12.3 Learning to code

A programming course, Comp Sci 101, can be taken either in-person, by attending a class at fixed days and times, or online, by doing sessions that can be taken at times the student chooses. The course coordinator wants to know whether students taking the course in these two different ways learn a different amount, as measured by their scores on the final exam for the course. This example comes from the before-times, so the final exam was taken in person by all students. The final exam was out of 45 marks. A total of 18 students took part in the study. Each student was allowed to choose the section they preferred. The data are in http://ritsokiguess.site/datafiles/proggo.csv.

Write a report of a complete and appropriate analysis of these data. Your report should include a description of the data in your own words, any necessary pre-processing steps, appropriate graphs, statistical analysis, assessment of assumptions for your preferred analysis, and a statement of conclusions. Imagine that your report will be read by the department Chair, who does not know about this study, and who still remembers some of their first-year Statistics course.

(My example report is in a later chapter, the one called Learning to Code.)

12.4 Treating dandruff

According to the Mayo Clinic, dandruff is "a common condition that causes the skin on the scalp to flake. It isn't contagious or serious. But it can be embarrassing and difficult to treat." Shampoos often claim to be effective in treating dandruff. In a study, four shampoos were compared:

- `PyrI`: 1% pyrithione zinc shampoo
- `PyrII`: the same as `PyrI` but with instructions to shampoo two times at each wash. The labels for these are `Pyr` with a Roman numeral I or II attached.

- `Keto`: 2% ketoconazole shampoo
- `Placebo`: a placebo shampoo

Each subject was randomly assigned to a shampoo. After six weeks of treatment, eight sections of the scalp were examined for each subject. Each section of the scalp was given a score that measured the amount of flaking on a scale of 0-10, less flaking being better. The response variable, called `Flaking`, was the sum of these eight scores, and is a whole number for each subject.

The data are in http://ritsokiguess.site/datafiles/dandruff.txt, with the data values separated by *tabs*.

Your task is to write a report on your analysis of this data set, and to make a recommendation for the best shampoo(s) out of the four studied here. The target audience for your report is the principal investigator of the study described above, who knows a lot about shampoo, but not so much about statistics. (They took a course some time ago that covered the material you've seen in this course so far, at about the level of STAB22 or STA 220.) Some things you might want to consider, in no particular order (you need to think about where and if to include these things):

- an Introduction, written in your own words as much as possible
- a Conclusion that summarizes what you found
- a suitable and complete piece of statistical inference
- a numerical summary of the data
- graph(s) of the data
- an assessment of the assumptions of your analysis
- citation of external sources
- anything else that you can make the case for including

In R Markdown (the text of an R Notebook), you can use `##` to make a heading (you can experiment with more or fewer `#` symbols).

Your aim is to produce a report, suitable for the intended audience, with all the important elements and no irrelevant ones, that is well-written and easy to follow. There is credit for good writing. For this report, you should include your code in with your report. (In a real report, you would probably show the output and not the code, but we are interested in your code here as well.)

(My example report is in a later chapter.)

My solutions follow. The example reports on coding and treating dandruff are separate chapters.

12.5 Atomic weight of carbon

The atomic weight of the chemical element carbon is 12. Two methods of measuring the atomic weight of samples of carbon were compared. The results are shown in link. The methods are labelled 1 and 2. The first task is to find out whether the two methods have different "typical" measures (mean or median, as appropriate) of the atomic weight of carbon.

For this question, compose a report in R Markdown. (R Markdown is what you use in an R Notebook, but you can also have a separate R Markdown document from which you can produce HTML, Word etc. output.) See part (a) for how to get this started.

Your report should read like an actual report, not just the answers to some questions that I set you. To help with that, write some text that links the parts of the report together smoothly, so that it reads as a coherent whole. The grader had 3 discretionary marks to award for the overall quality of your writing. The scale for this was:

- 3 points: excellent writing. The report flows smoothly, is easy to read, and contains everything it should (and nothing it shouldn't).
- 2 points: satisfactory writing. Not the easiest to read, but says what it should, and it looks at least somewhat like a report rather than a string of answers to questions.
- 1 point: writing that is hard to read or to understand. If you get this (or 0), you should consider what you need to do to improve when you write your project.
- 0 points: you answered the questions, but you did almost nothing to make it read like a report.

(a) Create a new R Markdown document. To do this, in R Studio, select File, New File, R Markdown. Type the report title and your name in the boxes, and leave the output on the default HTML. Click OK.

Solution

You'll see the title and your name in a section at the top of the document, and below that you'll see a template document, as you would for an R Notebook. The difference is that where you are used to seeing Preview, it now says "knit", but this has the same effect of producing the formatted version of your report.

■

(b) Write an introduction that explains the purpose of this study and the data collected in your own words.

Solution

Something like this:

> This study is intended to compare two different methods (labelled 1 and 2) for measuring the atomic weight of carbon (which is known in actual fact to be 12). Fifteen samples of carbon were used; ten of these were assessed using method 1 and the remaining five using method 2. The primary interest in this particular study is to see whether there is a difference in the mean or median atomic weight as measured by the two methods.

Before that, start a new section like this: `## Introduction`. Also, get used to expressing your understanding in your words, not mine. Using my words, in my courses, is likely to be worth very little.

■

(c) Begin an appropriately-titled new section in your report, read the data into R and display the results.

Solution

Values separated by spaces:

```
my_url <- "http://ritsokiguess.site/datafiles/carbon.txt"
carbon <- read_delim(my_url, " ")
```

```
Rows: 15 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): method, weight

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
carbon
```

```
# A tibble: 15 x 2
   method weight
    <dbl>  <dbl>
 1      1   12.0
 2      1   12.0
 3      1   12.0
 4      1   12.0
 5      1   12.0
 6      1   12.0
```

```
 7       1   12.0
 8       1   12.0
 9       1   12.0
10       1   12.0
11       2   12.0
12       2   12.0
13       2   12.0
14       2   12.0
15       2   12.0
```

I would expect you to include, without being told to include it, some text in your report indicating that you have sensible data: two methods labelled 1 and 2 as promised, and a bunch[1] of atomic weights close to the nominal figure of 12.

■

(d) Make an appropriate plot to compare the measurements obtained by the two methods. You might need to do something about the two methods being given as numbers even though they are really only identifiers. (If you do, your report ought to say what you did and why.)

Solution

The appropriate plot, with a categorical method and quantitative weight, is something like a boxplot. If you're not careful, `method` will get treated as a quantitative variable, which you don't want; the easiest way around that, for a boxplot at least, is to turn it into a factor like this:

```
ggplot(carbon, aes(x = factor(method), y = weight)) + geom_boxplot()
```

[1] It's probably better in a report to use language a bit more formal than *a bunch*. Something like *a number* would be better.

If you insist, you could do a faceted histogram (above and below, for preference):

```
ggplot(carbon, aes(x = weight)) + geom_histogram(bins = 5) +
  facet_wrap(~method, ncol = 1)
```

There are really not enough data values for a histogram to be of much help, so I don't like this as much.

If you are thinking ahead (we are going to be doing a t-test), then you'll realize that normality is the kind of thing we're looking for, in which case normal quantile plots would be the thing. However, we might have to be rather forgiving for method 2 since there are only 5 observations:

```
ggplot(carbon, aes(sample = weight)) +
  stat_qq() + stat_qq_line() +
  facet_wrap(~method)
```

I don't mind these coming out side by side, though I would rather have them squarer.

I would say, boxplots are the best, normal quantile plots are also acceptable, but expect to lose something for histograms because they offer only a rather crude comparison in this case.

■

(e) Comment briefly on what you see in your plot.

Solution

In boxplots, if that's what you drew, there are several things that deserve comment: the medians, the spreads and the shapes. The median for method 1 is a little bit lower than for

method 2 (the means are probably more different, given the shapes of the boxes). The spread for method 2 is a lot bigger. (Looking forward, that suggests a Welch-Satterthwaite rather than a pooled test.) As for shape, the method 2 measurements seem more or less symmetric (the whiskers are equal anyway, even if the position of the median in the box isn't), but the method 1 measurements have a low outlier. The histograms are hard to compare. Try to say something about centre and spread and shape. I think the method 2 histogram has a slightly higher centre and definitely bigger spread. On my histogram for method 1, the distribution looks skewed left. If you did normal quantile plots, say something sensible about normality for each of the two methods. For method 1, I would say the low value is an outlier and the rest of the values look pretty straight. Up to you whether you think there is a curve on the plot (which would indicate skewness, but then that highest value is too high: it would be bunched up with the other values below 12.01 if there were really skewness). For method 2, it's really hard to say anything since there are only five values. Given where the line goes, there isn't much you can say to doubt normality. Perhaps the best you can say here is that in a sample of size 5, it's difficult to assess normality at all.

■

(f) Carry out the most appropriate t-test. (You might like to begin another new section in your report here.)

Solution

This would be the Welch-Satterthwaite version of the two-sample t-test, since the two groups do appear to have different spreads:

```
t.test(weight ~ method, data = carbon)
```

```
    Welch Two Sample t-test

data:  weight by method
t = -1.817, df = 5.4808, p-value = 0.1238
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to
95 percent confidence interval:
 -0.027777288  0.004417288
sample estimates:
mean in group 1 mean in group 2
       12.00260        12.01428
```

Imagining that this is a report that would go to your boss, you ought to defend your choice of the Welch-Satterthwaite test (as I did above), and not just do the default t-test without comment.

If, in your discussion above, you thought the spreads were equal enough, then you should do the pooled t-test here, which goes like this:

```
t.test(weight ~ method, data = carbon, var.equal = T)
```

```
    Two Sample t-test

data:  weight by method
t = -2.1616, df = 13, p-value = 0.04989
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to
95 percent confidence interval:
 -2.335341e-02 -6.588810e-06
sample estimates:
mean in group 1 mean in group 2
       12.00260        12.01428
```

The point here is that you should do the right test based on your conclusion. Being consistent is the most important thing. (In this case, note that the P-values are very different. We'll get to that shortly.)

If we were doing this in SAS, as we see later, we'd get a test at the bottom of the output that compares the two variances. I feel that it's just as good to eyeball the spreads and make a call about whether they are "reasonably close". Or even, to always do the Welch-Satterthwaite test on the basis that it is pretty good even if the two populations have the same variance. (If this last point of view is one that you share, you ought to say something about that when you do your t-test.)

Extra: I guess this is a good place to say something about tests for comparing variances, given that you might be pondering that. There are several that I can think of, that R can do, of which I mention two.

The first is the F-test for variances that you might have learned in B57 (that is the basis for the ANOVA F-test):

```
var.test(weight ~ method, data = carbon)
```

```
    F test to compare two variances

data:  weight by method
F = 0.35768, num df = 9, denom df = 4, p-value = 0.1845
```

```
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.04016811 1.68758230
sample estimates:
ratio of variances
         0.3576842
```

This, unfortunately, is rather dependent on the data in the two groups being approximately normal. Since we are talking variances rather than means, there is no Central Limit Theorem to rescue us for large samples (quite aside from the fact that these samples are *not* large). Since the ANOVA F-test is based on the same theory, this is why normality is also more important in ANOVA than it is in a t-test.

The second is Levene's test. This doesn't depend on normality (at least, not nearly so much), so I like it better in general:

```r
library(car)
leveneTest(weight ~ factor(method), data = carbon)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  1  0.9887 0.3382
      13
```

Levene's test takes a different approach: first the absolute differences from the group medians are calculated, and then an ANOVA is run on the absolute differences. If, say, one of the groups has a larger spread than the other(s), its absolute differences from the median will tend to be bigger.[2] As for what we conclude here, well, neither of the variance tests show any significance at all, so from that point of view there is no evidence against using the pooled t-test. Having said that, the samples are small, and so it would be difficult to *prove* that the two methods have different variance, even if they actually did.[3]

Things are never as clear-cut as you would like. In the end, it all comes down to making a call and defending it.

■

(g) Do the most appropriate test you know that does not assume normally-distributed data.

[2]The use of absolute differences, and the median, downplays the influence of outliers. The assumption here is that the absolute differences from the medians are approximately normal, which seems a less big assumption than assuming the actual data are approximately normal.

[3]This is coming back to the *power* of something like Levene's test; the power of any test is not going to be very big if the sample sizes are small.

Solution

That would be Mood's median test. Since I didn't say anything about building it yourself, feel free to use `smmr`:

```
library(smmr)
median_test(carbon, weight, method)
```

```
$table
     above
group above below
    1     3     6
    2     4     1

$test
       what      value
1 statistic 2.80000000
2        df 1.00000000
3   P-value 0.09426431
```

As an aside, if you have run into a non-parametric test such as Mann-Whitney or Kruskal-Wallis that applies in this situation, be careful about using it here, because they have additional assumptions that you may not want to trust. Mann-Whitney started life as a test for "equal distributions".[4] This means that the null is equal location *and* equal spread, and if you reject the null, one of those has failed. But here, we suspect that equal spread will fail, so that the Mann-Whitney test may end up rejecting *whether or not* the medians are different, so it won't answer the question you want an answer to. Mood's median test doesn't have that problem; all it's saying if the null is true is that the medians are equal; the spreads could be anything at all.

The same kind of issues apply to the signed-rank test vs. the sign test. In the case of the signed-rank test, the extra assumption is of a symmetric distribution — to my mind, if you don't believe normality, you probably don't have much confidence in symmetry either. That's why I like the sign test and Mood's median test: in the situation where you don't want to be dealing with assumptions, these tests don't make you worry about that.

Another comment that you don't need to make is based on the not-quite-significance of the Mood test. The P-value is less than 0.10 but not less than 0.05, so it doesn't quite reach significance by the usual standard. But if you look up at the table, the frequencies seem rather unbalanced: 6 out of the remaining 9 weights in group 1 are below the overall median, but 4 out of 5 weights in group 2 are above. This seems as if it ought to be significant, but bear

[4] The test goes back to the 1940s.

in mind that the sample sizes are small, and thus Mood's median test needs *very* unbalanced frequencies, which we don't quite have here.

■

(h) Discuss the results of your tests and what they say about the two methods for measuring the atomic weight of carbon. If it seems appropriate, put the discussion into a section called Conclusions.

Solution

Begin by pulling out the P-values for your preferred test(s) and say what they mean. The P-value for the Welch-Satterthwaite t-test is 0.1238, which indicates no difference in mean atomic weights between the two methods. The Mood median test gives a similarly non-significant 0.0943, indicating no difference in the *median* weights. If you think both tests are plausible, then give both P-values and do a compare-and-contrast with them; if you think that one of the tests is clearly preferable, then say so (and why) and focus on that test's results.

If you thought the pooled test was the right one, then you'll have a bit more discussion to do, since its P-value is 0.0499, and at $\alpha = 0.05$ this test disagrees with the others. If you are comparing this test with the Mood test, you ought to make some kind of reasoned recommendation about which test to believe.

As ever, be consistent in your reasoning.

Extra: this dataset, where I found it, was actually being used to illustrate a case where the pooled and the Welch-Satterthwaite tests disagreed. The authors of the original paper that used this dataset (a 1987 paper by Best and Rayner;[5] the data come from 1924!) point out that the pooled t-test can be especially misleading when the smaller sample is also the one with the larger variance. This is what happened here.

In the Best and Rayner paper, the Mood (or the Mann-Whitney) test was not being considered, but I think it's good practice to draw a picture and make a call about which test is appropriate.

■

12.6 Sparrowhawks

(This is a report-writing question, but it also uses some regression techniques from later in the course.)

[5] Best, D. J., and J. C. W. Rayner. "Welch's Approximate Solution for the Behrens–Fisher Problem." Technometrics 29, no. 2 (May 1, 1987): 205–10. doi:10.1080/00401706.1987.10488211. The data set is near the end.

One of nature's patterns is the relationship between the percentage of adult birds in a colony that return from the previous year, and the number of new adults that join the colony. Data for 13 colonies of sparrowhawks can be found at link. The columns are the percentage of adults returning from the previous year, and the number of new adults that join the colony.

(a) Create a new R Markdown report, give it a suitable title, and ask for HTML output. Answer the questions that follow in your report. At any stage, you can Knit HTML to see how the report looks so far.

Solution

(Note: this is the previous version of Quarto, called R Markdown. The two are fairly similar.)

In R Studio, select File, New File, R Markdown. Fill in the Title, Author and leave the Default Output Format at HTML. You'll see a template report with the document info at the top. This is my document info:

```
---
title: "Sparrowhawks"
author: "Ken Butler"
date: "October 20, 2018"
output: html_document
---
```

This is known in the jargon as a "YAML block".[6] Below that is the template R Markdown document, which you can delete now or later.

■

(b) Read in the data and display the first few values. Add some text saying how many rows of data there are.

Solution

Read the data into a data frame. In your report, add some text like "we read in the data", perhaps after a section heading like "The data". Then add a *code chunk* by selecting Chunks and Insert Chunk, or by pressing control-alt-I. So far you have something like this.

[6]YAML stands for *Yet Another Markup Language*, but we're only using it in this course as the top bit of an R Markdown document.

```
1  ---
2  title: "Sparrowhawks"
3  author: "Ken Butler"
4  date: "October 20, 2018"
5  output: html_document
6  ---
7
8  ## The data
9
10 We begin by reading in the data.
11
12 ```{r}
13
14 ```
15
```

Inside the code chunk, that is, in the bit between the backtick characters, put R code, just as you would type it at the Console or put in an R notebook. In this case, that would be the following code, minus the message that comes out of `read_delim`:

```r
library(tidyverse)
my_url <- "http://ritsokiguess.site/datafiles/sparrowhawk.txt"
sparrowhawks <- read_delim(my_url, " ")
```

```
Rows: 13 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): returning, newadults

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
sparrowhawks
```

For you, it looks like this:

```
---
title: "Sparrowhawks"
author: "Ken Butler"
date: "October 20, 2018"
output: html_document
---

## The data

We begin by reading in the data, and loading in our usual stuff (for
use later):

```{r}
library(tidyverse)
my_url="http://www.utsc.utoronto.ca/~butler/c32/sparrowhawk.txt"
sparrowhawks=read_delim(my_url," ")
sparrowhawks
```

There are xxx rows of data altogether.
```

We don't know how many rows of data there are yet, so I've left a "placeholder" for it, when we figure it out. The file is annoyingly called `sparrowhawk.txt`, singular. Sorry about that. If you knit this (click on "Knit HTML" next to the ball of wool, or press control-shift-K), it should run, and you'll see a viewer pop up with the HTML output. Now you can see how many rows there are, and you can go back and edit the R Markdown and put in 13 in place of the x's, and knit again. You might be worried about how hard R is working with all this knitting. Don't worry about that. R can take it. Mine looked like this:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/sparrowhawk.txt"
sparrowhawks=read_delim(my_url," ")
```

```
## Parsed with column specification:
## cols(
##   returning = col_integer(),
##   newadults = col_integer()
## )
```

```
sparrowhawks
```

```
## # A tibble: 13 x 2
##    returning newadults
##        <int>     <int>
##  1        74         5
##  2        66         6
##  3        81         8
##  4        52        11
##  5        73        12
##  6        62        15
##  7        52        16
##  8        45        17
##  9        62        18
## 10        46        18
## 11        60        19
## 12        46        20
## 13        38        20
```

There are 13 rows of data altogether, with a value of the number of returning and new adults for each one.

There is a better way of adding values that come from the output, which I mention here in case you are interested (if you are not, feel free to skip this). What you do is to make what is called an "inline code chunk". Where you want a number to appear in the text, you have some R Markdown that looks like this:

```
19
20  There are `r nrow(sparrowhawks)` rows altogether.
21
```

The piece inside the backticks is the letter `r`, a space, and then one line of R code. The one line of code will be run, and all of the stuff within the backticks will be replaced in the output by the result of running the R code, in this case the number 13. Typically, you are extracting a number from the data, like the number of rows or a mean of something. If it's a decimal number, it will come out with a lot of decimal places unless you explicitly `round` it. OK, let me try it: the data frame has 13 rows altogether. I didn't type that number; it was calculated from the data frame. Woo hoo!

■

(c) Create a new section entitled "Exploratory analysis", and create a scatterplot for predicting number of new adults from the percentage of returning adults. Describe what you see, adding some suitable text to your report.

Solution

The R code you add should look like this, with the results shown (when you knit the report again):

```
library(tidyverse)
ggplot(sparrowhawks, aes(x = returning, y = newadults)) +
  geom_point() + geom_smooth()
```

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

The piece of report that I added looks like this:

```
## Exploratory analysis

We should look at a scatterplot of the data, to see what, if any,
relationship there is. We add a smooth trend, to guide the eye:

```{r}
ggplot(sparrowhawks,aes(x=returning,y=newadults))+
  geom_point()+geom_smooth()
```

This seems to be something of a downward trend, though it is not
completely convincing. That is, a larger percentage of returning
adults is associated with a *smaller* number of new adults. Is the
relationship straight? Well, the smooth trend has a strange
wiggle in the middle, but the trend overall does not seem
obviously curved.
```

Note (i) that you have to do nothing special to get the plot to appear, and (ii) that I put "smaller" in italics, and you see how.

■

(d) Obtain the correlation between the two variables. Is this consistent with the scatterplot? Explain briefly. (The R function you need is `cor`. You can feed it a data frame.)

Solution

The appropriate R code is this, in another code chunk:

```r
with(sparrowhawks, cor(newadults, returning))
```

```
[1] -0.7484673
```

Or you can ask for the correlations of the whole data frame:

```r
cor(sparrowhawks)
```

```
           returning  newadults
returning  1.0000000 -0.7484673
newadults -0.7484673  1.0000000
```

This latter is a "correlation matrix" with a correlation between each column and each other column. Obviously the correlation between a column and itself is 1, and that is *not* the one we want.

I added this to the report (still in the Exploratory Analysis section, since it seems to belong there):

```
What is the correlation between these two variables?

```{r}
with(sparrowhawks,cor(newadults,returning))
```

The correlation is about -0.75. This seems surprisingly close to -1 to me, since I
didn't think the trend was that strong, but it is no surprise that the correlation
is negative, since the trend is definitely downward rather than
upward.
```

■

(e) Obtain the regression line for predicting the number of new adults from the percentage of returning adults.

Solution

This R code, in another code chunk:

```r
newadults.1 <- lm(newadults ~ returning, data = sparrowhawks)
summary(newadults.1)
```

```
Call:
lm(formula = newadults ~ returning, data = sparrowhawks)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.8687 -1.2532  0.0508  2.0508  5.3071 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 31.93426    4.83762   6.601 3.86e-05 ***
returning   -0.30402    0.08122  -3.743  0.00325 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 3.667 on 11 degrees of freedom
Multiple R-squared:  0.5602,    Adjusted R-squared:  0.5202
F-statistic: 14.01 on 1 and 11 DF,  p-value: 0.003248
```

■

(f) What are the intercept and slope of your regression line? Is the slope significant? What does that mean, in the context of the data?

Solution

See the output in the previous part. That's what we need to talk about. I added this to the report. I thought we deserved a new section here:

```
## Regression analysis


The regression for predicting the number of new adults from the
percentage of returning adults is as shown:

```{r}
newadults.1=lm(newadults~returning,data=sparrowhawks)
summary(newadults.1)
```

The intercept is about 32 and the slope is about -0.3. This means that
each additional one percentage point in returning adults is associated
with a *decrease* of approximately 0.3 new adults.

The P-value attached to the slope is about 0.003. This means that the
slope is *not* zero, and that there is a significant association
between the number of returning adults and the number of new
adults. (This is despite the apparently weak trend, and the small
number, 13, of data points.)
```

■

(g) Create a scatterplot of the data with the regression line on it.

Solution

This code. Using `geom_smooth` with `method="lm"` will add the regression line to the plot:

```r
ggplot(sparrowhawks, aes(x = returning, y = newadults)) +
  geom_point() + geom_smooth(method = "lm")
```

```
`geom_smooth()` using formula = 'y ~ x'
```

I added a bit of text to the report, which I will show in a moment.

■

(h) For short-lived birds, the association between these two variables is positive: changes in weather and food supply cause the populations of new and returning birds to increase together. For long-lived territorial birds, however, the association is negative because returning birds claim their territories in the colony and do not leave room for new recruits. Which type of species is the sparrowhawk? Add a short Conclusions section to your report with discussion of this issue.

Solution

My addition to the report looks like this:

```

A plot of the data with the regression line superimposed is as shown:

```{r}
ggplot(sparrowhawks,aes(x=returning,y=newadults))+
  geom_point()+geom_smooth(method="lm")
```

Conclusions

The association is negative, as seen from the scatterplot and
correlation, and also from the negative slope of the regression
line. This suggests that sparrowhawks are long-lived territorial
birds rather than short-lived birds. We would expect to observe
returning birds displaying territorial behaviour that would
discourage new adults from joining the colony.

```

I think that rounds off the report nicely.

■

# 13 Learning to code

```
library(tidyverse)
```

A programming course, Comp Sci 101, can be taken either in-person, by attending a class at fixed days and times, or online, by doing sessions that can be taken at times the student chooses. The course coordinator wants to know whether students taking the course in these two different ways learn a different amount, as measured by their scores on the final exam for the course. This example comes from the before-times, so the final exam was taken in person by all students. The final exam was out of 45 marks. A total of 18 students took part in the study. Each student was allowed to choose the section they preferred. The data are in http://ritsokiguess.site/datafiles/proggo.csv.

Write a report of a complete and appropriate analysis of these data. Your report should include a description of the data in your own words, any necessary pre-processing steps, appropriate graphs, statistical analysis, assessment of assumptions for your preferred analysis, and a statement of conclusions. Imagine that your report will be read by the department Chair, who does not know about this study, and who still remembers some of their first-year Statistics course.

Solution

My report follows. The occasional appearance of "(see note x)" from time to time indicates some extra discussion, at the end. Ignore those on your first read-through. Your report doesn't have to be the same as mine, but it should be complete and coherent, and readable by someone with a first course in Statistics.

13.1 Introduction

Do students learn programming more or less effectively from an online course, completed on their own time, compared with a regular in-person lecture course that meets at the same times every week? In Comp Sci 101, a study was carried out in which 18 students, 9 each in the online and in-person sections, were assessed for learning by means of the course final exam (out of 45 marks). We compare the mean final exam scores for the students in the two sections.

13.2 Data and pre-processing

We begin by reading in the data:

```
library(tidyverse)
```

```
my_url <- "http://ritsokiguess.site/datafiles/proggo.csv"
prog0 <- read_csv(my_url)
```

```
Rows: 9 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (2): online, classroom

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
prog0
```

```
# A tibble: 9 x 2
  online classroom
   <dbl>     <dbl>
1     32        35
2     37        31
3     35        29
4     28        25
5     41        34
6     44        40
7     35        27
8     31        32
9     34        31
```

The nine students in each section are in separate columns. This is not an appropriate layout for analysis because each row contains data for *two* separate, unrelated students, and we need to have each student's score in its own row. (See note 1.) This means making the data longer (see note 2):

```
prog0 %>% pivot_longer(everything(),
  names_to = "instruction",
```

```
    values_to = "mark") -> prog
  prog
```

```
# A tibble: 18 x 2
   instruction  mark
   <chr>       <dbl>
 1 online         32
 2 classroom      35
 3 online         37
 4 classroom      31
 5 online         35
 6 classroom      29
 7 online         28
 8 classroom      25
 9 online         41
10 classroom      34
11 online         44
12 classroom      40
13 online         35
14 classroom      27
15 online         31
16 classroom      32
17 online         34
18 classroom      31
```

Now everything is arranged as we need, and we can proceed to analysis.

13.3 Analysis

We begin by visualizing the data. With one quantitative variable `mark` and one categorical variable `instruction`, a boxplot will give us an overall picture (see note 3):

```
ggplot(prog, aes(x= instruction, y = mark)) + geom_boxplot()
```

It looks as if the average (median) mark is somewhat higher for the students in the online section. In addition, both distributions look reasonably symmetric with no outliers, and they appear to have similar spread. (see note 4.)

With that in mind, it seems sensible to compare the mean final exam marks in the two groups by a pooled two-sample t-test, in order to see whether the apparent difference in performance is any more than chance. The aim of the course coordinator was to see whether there was any difference between the two sections, without having any prior idea about which teaching method would be better, so a two-sided test is appropriate: (see note 5.)

```
t.test(mark~instruction, data = prog, var.equal = TRUE)
```

```
    Two Sample t-test

data:  mark by instruction
t = -1.6495, df = 16, p-value = 0.1185
alternative hypothesis: true difference in means between group classroom and group online is
95 percent confidence interval:
 -8.379039  1.045706
sample estimates:
mean in group classroom    mean in group online
               31.55556                35.22222
```

The P-value of 0.1185 is not smaller than 0.05, so there is no evidence of any difference between the mean final exam marks of the students in the two sections. The difference between the two groups shown on the boxplot is the kind of thing that could be observed by chance if the students were performing equally well under the two methods of instruction. (See notes 6 and 7.)

This t-test comes with the assumption that the data in each group come from a normal distribution, at least approximately. With two small samples, this will not be easy to assess, but we can at least look for any gross violations, using a normal quantile plot for each group:

```
ggplot(prog, aes(sample = mark)) + stat_qq() + stat_qq_line() +
facet_wrap(~instruction)
```

The largest observation in the classroom group and the largest two observations in the online group are a little bigger than expected, but they were not big enough to be flagged as outliers on the boxplots, so I conclude that the normality is good enough to justify the t-test that we did. (See note 8.)

13.4 Conclusions (see note 9)

We found that there is no significant difference in the performance of the students learning in the classroom compared to those learning online. Before leaping to generalize to other classes,

however, we should note two limitations of the study (see note 10):

- the sample sizes were very small; if we had observed this size of difference between the two groups in larger samples, we might have been able to show that the difference was significant.
- we have no information about how the students were allocated to the groups, and it seems likely that the students were allowed to choose their own method of instruction. If the students had been randomly allocated to instruction method, we could have been more confident that any differences observed were due to the instruction method, rather than also having something to do with the relative ability of the students who chose each instruction method.

We feel that it would be worth running another study of this type, but with larger sample sizes and randomly allocating students to instruction types. This latter, however, risks running into ethical difficulties, since students will normally wish to choose the section they are in.

Thus ends the report.

Notes:

1. Say something about the kind of data layout you have, and whether it's what you want.

2. You can do a t-test without rearranging the data (the method is that of the "all possible t-tests" discussion in the ANOVA section), but if you try to draw plots with the data laid out that way, you will at best be repeating yourself a lot and at worst get stuck (if you try to make side-by-side boxplots). The right format of data should give you no surprises!

I split the pivot-longer onto several lines so that it wouldn't run off the right side of the page. Recall that R doesn't mind if you have it on one line or several, as long as it can tell that the current line is incomplete, which it must be until it sees the closing parenthesis on the `pivot_longer`.

3. A normal quantile plot is also justifiable here, but I think it would take more justification, because you would have to sell your reader on the need for normality this early in the story. A standard plot like a boxplot needs no such justification; it just describes centre, spread and shape, exactly the kind of thing your first look at the data should be telling you about.

4. Say something about what the graph is telling you. It makes sense to do as I did and look forward to the kind of inference you are going to try. If you do a normal quantile plot here, you can formally assess the assumptions before you do the test, which you might argue makes more sense, rather than doing a second plot and assessing the assumptions later as I do.

5. You might be able to justify a one-sided test here, along the lines of "is the online instruction significantly *worse*?", but in any case, you need to justify the one-sided or two-sided test that you do.
6. Conclusion *in the context of the data*, as ever. Writing "we fail to reject the null hypothesis" and then stopping invites your reader to ask "so what?".
7. You might have chosen to do a different test, but your choice needs to be properly justified. I think the pooled t-test is the best choice here. If you thought those marks were not normal enough, then you need to do Mood's median test, explaining why. That comes out like this:

```
median_test(prog, mark, instruction)
```

```
$table
           above
group       above below
  classroom     3     6
  online        6     3

$test
       what     value
1 statistic 2.0000000
2        df 1.0000000
3   P-value 0.1572992
```

The (two-sided) P-value is a little bigger than the one for the pooled t-test, but the conclusion is the same. (If you think the test should be one-sided, justify dividing your P-value by 2, if you can. In the case that your one-sided alternative was that classroom teaching was *better*, you cannot reject the null in favour of that, because the online students actually have better marks in these samples. In that case, halving the P-value would not be justifiable.)

If you thought that the normality was all right, but the equal spreads was not, you will need to justify the unequal spreads. Perhaps the best way here is to say that you are unsure whether the spreads are close enough to equal, so you are doing the Welch test to be safe. That looks like this:

```
t.test(mark~instruction, data = prog)
```

```
    Welch Two Sample t-test
```

```
data:  mark by instruction
t = -1.6495, df = 15.844, p-value = 0.1187
alternative hypothesis: true difference in means between group classroom and group online is
95 percent confidence interval:
 -8.382820  1.049486
sample estimates:
mean in group classroom    mean in group online
               31.55556                35.22222
```

As you see, the P-values of the pooled and Welch t-tests are almost identical (which is what I was guessing), but to do the Welch test without comment reveals that you are *not thinking* about which of the two tests is more appropriate here. In the real world, you might get away with it (for these data, the conclusions are the same), but in this course I need to see that your thought process is correct.

A final observation in this note: all three tests give you similar P-values and the same conclusion, so that in the end it didn't really matter which one of them you did. When that happens, it's usually (as here) a sign that a t-test will be best, because it makes the best use of the data (and thus you will get the most power in your test). I cannot stop you doing all three tests behind the scenes and using this to help decide, but strictly speaking your P-value will not be what you say it is, because these tests (really, any tests) are designed so that the test you choose is the only one you do. In any case, your *report* should include only the *one* test you thought was most appropriate. Your reader does not have the time or patience for a detailed comparison of the three tests.

8. You might have combined the assessment of assumptions with your first plot, particularly if you chose a normal quantile plot instead of a boxplot there. As long as you have assessed normality (and equal spreads, if you are happy with the normality) somewhere, this is fine. In particular, if you ended up doing a Mood median test, you have presumably already decided that the normality was not good enough, and (I hope) you already discussed that somewhere.

Again, bear in mind who is (in the setup that we have) reading your report: someone who might remember about t-tests and normality. Ideas like the bootstrap are *far* too advanced to go into a report like this. If you must include something like that, you need to put it in an appendix, not the main body of the report, so that the person reading your report can get the main ideas without having to wade through that.

The rest of this note is an Extra:

Here is a slightly different way to approach the bootstrap in this case, that takes care of assessing the normality of both groups at once.

The first step is `nest_by`, which does two things, one of which is invisible:

```r
prog %>% nest_by(instruction)
```

```
# A tibble: 2 x 2
# Rowwise:  instruction
  instruction                data
  <chr>       <list<tibble[,1]>>
1 classroom               [9 x 1]
2 online                  [9 x 1]
```

This (visibly) creates a list-column containing two mini dataframes `data`. They are the original data with all the data except `instruction` in each one: that is, the other column `mark`. The top one is the nine marks for the students in the `classroom` section, and the bottom one is the nine marks for the students in the `online` section. The invisible thing is that `nest_by` includes a `rowwise`, so that what we do after this is one for *each* row, one at a time.

What we do next is to generate a lot of bootstrap samples, for each group. First, a vector of simulation numbers, one for each method of instruction. This only needs to be said once because `nest_by` is like `group_by`:

```r
prog %>% nest_by(instruction) %>%
  mutate(sim = list(1:10))
```

```
# A tibble: 2 x 3
# Rowwise:  instruction
  instruction                data sim
  <chr>       <list<tibble[,1]>> <list>
1 classroom               [9 x 1] <int [10]>
2 online                  [9 x 1] <int [10]>
```

Then we need to `unnest` those `sim`s, so that we can put a bootstrap sample next to each one:

```r
prog %>% nest_by(instruction) %>%
  mutate(sim = list(1:10)) %>%
  unnest(sim)
```

```
# A tibble: 20 x 3
# Groups:   instruction [2]
   instruction                data   sim
   <chr>       <list<tibble[,1]>> <int>
 1 classroom               [9 x 1]     1
```

```
 2 classroom              [9 x 1]     2
 3 classroom              [9 x 1]     3
 4 classroom              [9 x 1]     4
 5 classroom              [9 x 1]     5
 6 classroom              [9 x 1]     6
 7 classroom              [9 x 1]     7
 8 classroom              [9 x 1]     8
 9 classroom              [9 x 1]     9
10 classroom              [9 x 1]    10
11 online                 [9 x 1]     1
12 online                 [9 x 1]     2
13 online                 [9 x 1]     3
14 online                 [9 x 1]     4
15 online                 [9 x 1]     5
16 online                 [9 x 1]     6
17 online                 [9 x 1]     7
18 online                 [9 x 1]     8
19 online                 [9 x 1]     9
20 online                 [9 x 1]    10
```

Now we draw a bootstrap sample from the `data` to the left of each `sim` (the ones next to `classroom` are the in-person ones, and the ones next to `online` are the online ones, so we will be taking bootstrap samples of the right thing). This here is where the `rowwise` goes:

```
prog %>% nest_by(instruction) %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(the_sample = list(sample(data$mark, replace = TRUE))) %>%
  mutate(the_mean = mean(the_sample)) %>%
  ggplot(aes(sample = the_mean)) + stat_qq() +
    stat_qq_line() + facet_wrap(~instruction)
```

These are close to their lines, which tells me indeed that the two-sample t-test I did was perfectly reasonable.

As I say, though, this is very much for *you*, and not for the report, given who would be reading it. If you were doing a presentation on this, the bootstrap stuff is something you would keep in reserve in case someone asks about the appropriateness of the t-test, and *then* you could talk about it, but otherwise you wouldn't mention it at all.

9. You definitely need some conclusions. If the department chair is busy (quite likely), this is the only part of the entire report they may be able to read.

10. This is the place to put limitations, and recommendations for next time. The sample size one is (I hope) obvious, but you can also get at the other one by asking yourself how the students were assigned to instruction methods. The classical comparative study assigns them at random; that way, you know that the two groups of students are at least supposed to be about equal on ability (and anything else) before you start. But if the students choose their own groups, it might be that (say) the weaker students choose the classroom instruction, and in that case the reason the online group came out looking better might be that they were stronger students: it could have nothing to do with the effectiveness of the learning environment at all.

■

14 Treating dandruff

14.1 Introduction

Shampoos are often claimed to be effective at treating dandruff. In a study, the dandruff-treating properties of four shampoos were compared. These four shampoos were, as referred to in the dataset:

- `PyrI`: 1% pyrithione zinc shampoo
- `PyrII`: as `PyrI` but with instructions to shampoo two times at each wash.[1]
- `Keto`: 2% ketoconazole shampoo
- `Placebo`: a placebo shampoo

Each of the experimental subjects was randomly given one of the shampoos. After using their shampoo for six weeks, eight sections of the subject's scalp were examined for each subject. Each section of the scalp was given a score that measured the amount of flaking on a scale of 0-10. The response variable, called `Flaking`, was the sum of these eight scores, and is a whole number for each subject. A smaller value of `Flaking` indicates less dandruff.[2]

Our aim is to see which shampoo or shampoos are most effective at treating dandruff, that is, have the smallest value of `Flaking` on average.

14.2 Exploratory analysis

We begin by reading in the data:

```
my_url <- "http://ritsokiguess.site/datafiles/dandruff.txt"
dandruff <- read_tsv(my_url)
```

[1]The piece in the problem statement about why these two labels were used is clarification for you and doesn't belong in the report. If you leave it in, you need to at least paraphrase it; simply copying it without having a reason to do so shows that you are not thinking.

[2]I'm using a fair few of my own words from the question. This is OK if you think they are clear, but the aim is to write a report that sounds like you rather than me.

```
Rows: 355 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): Treatment
dbl (3): OBS, GroupNum, Flaking

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
dandruff
```

```
# A tibble: 355 x 4
     OBS Treatment GroupNum Flaking
   <dbl> <chr>        <dbl>   <dbl>
 1     1 PyrI             1      17
 2     2 PyrI             1      16
 3     3 PyrI             1      18
 4     4 PyrI             1      17
 5     5 PyrI             1      18
 6     6 PyrI             1      16
 7     7 PyrI             1      17
 8     8 PyrI             1      20
 9     9 PyrI             1      17
10    10 PyrI             1      17
# i 345 more rows
```

355 subjects took part in the study altogether. The shampoo used is indicated in the `Treatment` column. The remaining columns `OBS` and `GroupNum` will not be used in this analysis.

Numerical summaries of the data are as shown:

```r
dandruff %>% group_by(Treatment) %>%
summarise(n=n(), fl_mean=mean(Flaking), fl_sd=sd(Flaking))
```

```
# A tibble: 4 x 4
  Treatment     n fl_mean fl_sd
  <chr>     <int>   <dbl> <dbl>
1 Keto        106    16.0 0.931
2 Placebo      28    29.4 1.59
3 PyrI        112    17.4 1.14
4 PyrII       109    17.2 1.35
```

There are approximately 100 observations in each group, apart from the Placebo group, which had only 28. The mean number of flakes is much higher for the Placebo group than for the others, which seem similar. The group standard deviations are fairly similar.

With a categorical `Treatment` and a quantitative `Flakes`, a suitable graph is a side-by-side boxplot:

```
ggplot(dandruff, aes(x=Treatment, y=Flaking)) + geom_boxplot()
```

Once again, we see that the flaking for the Placebo shampoo is much higher than for the others. There are outliers in the `PyrI` group, but given that the data values are all whole numbers, they are not far different from the rest of the data. Considering these outliers, the spreads of the groups all look fairly similar and the distributions appear more or less symmetric.[3]

14.3 Analysis of Variance

For comparing four groups, we need some kind of analysis of variance. Having seen that the `Flaking` values within the four groups are more or less normal with more or less equal spreads, we run a standard ANOVA:

[3]Offer supported opinions of your own here, which don't need to be the same as mine. Alternatively, you can get the graph and numerical summaries first and comment on them both at once.

```r
dandruff.1 <- aov(Flaking~Treatment, data=dandruff)
summary(dandruff.1)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)    
Treatment     3   4151  1383.8   967.8 <2e-16 ***
Residuals   351    502     1.4                   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With an extremely small P-value, we conclude that the four shampoos do not all have the same mean value of `Flaking`.

To find out which ones are different from which, we use Tukey's method:

```r
TukeyHSD(dandruff.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Flaking ~ Treatment, data = dandruff)

$Treatment
                     diff         lwr         upr     p adj
Placebo-Keto   13.3645553  12.7086918  14.0204187 0.0000000
PyrI-Keto       1.3645553   0.9462828   1.7828278 0.0000000
PyrII-Keto      1.1735330   0.7524710   1.5945950 0.0000000
PyrI-Placebo  -12.0000000 -12.6521823 -11.3478177 0.0000000
PyrII-Placebo -12.1910223 -12.8449971 -11.5370475 0.0000000
PyrII-PyrI     -0.1910223  -0.6063270   0.2242825 0.6352706
```

All of the shampoo treatments are significantly different from each other except for the two pyrithione ones. To see which shampoos are best and worst, we remind ourselves of the treatment means:

```r
dandruff %>% group_by(Treatment) %>%
summarise(n=n(), fl_mean=mean(Flaking), fl_sd=sd(Flaking))
```

```
# A tibble: 4 x 4
  Treatment     n fl_mean fl_sd
  <chr>     <int>   <dbl> <dbl>
```

```
1 Keto        106    16.0 0.931
2 Placebo      28    29.4 1.59
3 PyrI        112    17.4 1.14
4 PyrII       109    17.2 1.35
```

The placebo shampoo has significantly more flaking than all the others, and the ketoconazole[4] shampoo has significantly less flaking than all the others. From this analysis, therefore, we would recommend the ketoconazole shampoo over all the others.

14.4 Assessment of Assumptions

The analysis of variance done above requires that the observations within each treatment group (shampoo) be approximately normally distributed, given the sample sizes, with approximately equal spreads. To assess this, we look at normal quantile plots[5] for each shampoo:[6]

```
ggplot(dandruff, aes(sample=Flaking)) + stat_qq() + stat_qq_line() +
facet_wrap(~Treatment, scales = "free")
```

[4]Use the full name of the shampoo if you are making a conclusion about it.

[5]I've used `scales = "free"` to get the plots to fill their boxes, for the best assessment of normality. The downside of doing it this way is that you cannot use the slopes of the lines to compare spreads. I think this way is still better, though, because the mean for placebo is so much bigger than the others that if you use the same scale for each plot, you'll be wasting a lot of plot real estate that you could use to get a better picture of the normality.

[6]It's also good, arguably clearer, to use this as your exploratory plot. This enables you to get to a discussion about normality earlier and you might decide in that case that you don't even need this discussion. You can do the assessment of assumptions first, and then do the corresponding analysis, or you can pick an apparently reasonable analysis and then critique it afterwards. Either way is logical here. In other cases it might be different; for example, in a regression, you might need to fit a model first and improve it after, since it may not be so clear what a good model might be off the top.

Given that the data are whole numbers, the distributions each appear close to the lines, indicating that the distributions are close to normal in shape.[7] The distribution of the `PyrI` values is slightly long-tailed, but with over 100 observations in that group, this shape is not enough to invalidate the normality assumption.[8]

Having concluded that the normality is sufficient, we need to assess the equality of spreads. Referring back to our summary table:

```
dandruff %>% group_by(Treatment) %>%
summarise(n=n(), fl_mean=mean(Flaking), fl_sd=sd(Flaking))
```

```
# A tibble: 4 x 4
  Treatment     n fl_mean fl_sd
  <chr>     <int>   <dbl> <dbl>
1 Keto        106    16.0 0.931
2 Placebo      28    29.4 1.59
3 PyrI        112    17.4 1.14
```

[7]This is a way to write it if you suspect your reader won't remember what a normal quantile plot is, and by writing it this way you won't insult their intelligence if they *do* remember after all. The other side benefit of writing it this way is that it shows *your* understanding as well.

[8]If you have any doubts about sufficient normality, you need to make sure you have also considered the relevant sample size, but if you are already happy with the normality, there is no need. The placebo group, for example, is the smallest, but its shape is if anything short-tailed, so its non-normality will be no problem no matter how small the sample is.

```
4 PyrII       109    17.2 1.35
```

we note that the spreads are not greatly different, and so the equal-spread assumption appears to be satisfied.[9]

In summary, the assumptions for the analysis of variance we did seem to be reasonably well satisfied, and we can have some confidence in our conclusions.

14.5 Conclusions

We found that the ketoconazole shampoo produced the smallest mean flaking, and its mean was significantly smaller than that of all the other shampoos. This shampoo can be recommended over the others. There was no significant difference between the two pyrithione treatments; shampooing twice had no benefit over shampooing once.

The difference in means between the ketoconazole and the two pyrithione shampoos was only about 1.2. This difference was significant because of the large sample sizes, but it is a separate question as to whether a difference of this size is of practical importance. If it is not, any of the shampoos except for the placebo can be recommended.

14.6 End

That is the end of my report. You, of course, don't need the word "end" or any of the footnotes I had. These were to draw your attention to other things that don't necessarily belong in the report, but I would like you to be aware of them. When you reach the end of your report, you can just stop.[10]

Some extra, bigger, thoughts (there are quite a few of these. I hope I don't repeat things I also say in the "sidenotes"):

Extra 1: The placebo group is much smaller than the others, but all the groups are pretty big by ANOVA standards. Apparently what happened is that originally the three "real" treatments had 112 subjects each, but the placebo had 28 (ie., a quarter of the subjects that the other groups had), and a few subjects dropped out. There's no problem, in one-way ANOVAs of this kind, with having groups of unequal sizes; the F-test is fine, and as long as you use a suitable extension of Tukey that deals with unequal sample sizes, you are OK there too. `TukeyHSD`, according to the help file, "incorporates an adjustment for sample size that produces sensible intervals for mildly unbalanced designs". In this case, we might be holding our breath a bit,

[9] I'd rather assess equality of spreads by eyeballing them than by doing a test, but if you really want to, you could use Levene's test, illustrated elsewhere in PASIAS and in Extra 4 for these data. It works for any number of groups, not just two.

[10] I wanted to make it clear where my report ended and where the additional chat began.

depending on what "mildly unbalanced" actually means. Usually in this kind of study, you have the groups about the same size, because proving that the smallest group differs from any of the others is more difficult. I guess these researchers were pretty confident that the placebo shampoo would be clearly worse than the others! (Additional: `TukeyHSD` uses the so-called Tukey-Kramer test when sample sizes within groups are unequal. My understanding is that this is good no matter what the sample sizes are.)

Extra 2: The mean for `Placebo` is quite a lot bigger than for the other groups, so a plot with different scales for each facet is best. Otherwise you get this kind of thing, which is much harder to read:

```
ggplot(dandruff, aes(sample=Flaking)) + stat_qq() + stat_qq_line() +
facet_wrap(~Treatment)
```

The points fill less than half their facets, which makes the plots harder to understand. This also makes it look as if the distributions are more normal, because the vertical scale has been compressed. Having a better basis for assessing the normality is a good idea, given that the purpose of the plot *is* assessing the normality! Hence, using `scales = "free"` is best.

Extra 3: You might have been wondering why the boxplots, which are the usual thing in these circumstances, look worse than the normal quantile plots.

Let's revisit them and see what happened:

```
ggplot(dandruff, aes(x=Treatment, y=Flaking)) + geom_boxplot()
```

The `Placebo` group has the largest IQR, and the `PyrI` group appears to have two outliers. We need to bear in mind, though, that the data values are whole numbers and there might be repeats; also, what looks like an outlier here might not look quite so much like one when we see all the data.

What you can do is to add a `geom_point` on to the plot to plot the observations as points:

```
ggplot(dandruff, aes(x=Treatment, y=Flaking)) +
geom_boxplot() + geom_point()
```

But there are a lot more data points than this! What happened is that a point at, say, 20 is *all* the observations in that group that were 20, of which there might be a lot, but we cannot see how many, because they are printed on top of each other. To see all the observations, we can **jitter** them: that is, plot them all *not* in the same place. In this case, we have the whole width of the boxplot boxes to use; we could also jitter vertically, but I decided not to do that here. There is a `geom_jitter` that does exactly this:[11]

```
ggplot(dandruff, aes(x=Treatment, y=Flaking)) + geom_boxplot() +
geom_jitter(height = 0)
```

[11] I explain the `height=0` below the plot.

The plot is rather messy,[12] but now you see everything. The `height=0` means not to do any vertical jittering: just spread the points left and right.[13] Where the points are exactly on the x-scale is now irrelevant; this is just a device to spread the points out so that you can see them all.

I left the vertical alone so that you can still see the actual data values. Even though the highest and lowest values in `PyrI` were shown as outliers on the original boxplot, you can see that they are really not. When the data values are discrete (separated) like this, an apparent outlier may be only one bigger or smaller than the next value, and thus not really an outlier at all.

To try the vertical jittering too, use the defaults on `geom_jitter`:

```
ggplot(dandruff, aes(x=Treatment, y=Flaking)) + geom_boxplot() + geom_jitter()
```

[12]It looks to me as if the boxplot has been attacked by mosquitoes.

[13]The default jittering is up to a maximum of not quite halfway to the next value. Here that means that each observation is nearest to the box it belongs with.

This maybe spreads the points out better, so you can be more certain that you're seeing them all, but you lose the clear picture of the data values being whole numbers.

When Tukey popularized the boxplot, his idea was that it would be drawn by hand with relatively small samples, and when you draw boxplots for large samples, you can get an apparently large number of outliers, that are not in retrospect quite as extreme as they may look at first. This may also have happened here. Tukey, however, did not invent the boxplot; credit for that goes to Mary Eleanor Spear with her "range plot".[14]

Extra 4: More discussion of normality. The assumptions for a standard ANOVA (that is to say, not a Welch ANOVA) are normally-distributed data within each treatment group, with equal spreads. What that means in practice is that you want normal *enough* data given the sample sizes, and approximately equal spreads. My normal quantile plots are a long way back, so let's get them again:

```
ggplot(dandruff, aes(sample=Flaking)) + stat_qq() + stat_qq_line() +
facet_wrap(~Treatment, scales = "free")
```

[14] I learned this today.

The data values are all whole numbers, so we get those horizontal stripes of `Flaking` values that are all the same. As long as these more or less hug the line, we are all right. The `PyrII` values certainly do. In my top row, `Keto` and `Placebo` are not quite so good, but they have *short* tails compared to the normal, so there will be no problem using the means for these groups, as ANOVA does. The only one that is problematic at all is `PyrI`. That has slightly long tails compared to a normal. (You could, I suppose, call those highest and lowest values "outliers", but I don't think they are far enough away from the rest of the data to justify that.) Are these long tails a problem? That depends on how many observations we have:

```
dandruff %>% group_by(Treatment) %>%
summarise(n=n(), mean_flaking=mean(Flaking), sd_flaking=sd(Flaking))
```

```
# A tibble: 4 x 4
  Treatment     n mean_flaking sd_flaking
  <chr>     <int>        <dbl>      <dbl>
1 Keto        106         16.0      0.931
2 Placebo      28         29.4      1.59
3 PyrI        112         17.4      1.14
4 PyrII       109         17.2      1.35
```

There are 112 of them. Easily enough to overcome those long tails. So, to my mind, normality is no problem.

Aside: you might be wondering whether you can make nicer-looking tables in your reports. There are several ways. The `gt` package is the most comprehensive one I know, and has links to a large number of others (at the bottom of its webpage). The simplest one I know of is `kable` in the `knitr` package. You may well have that package already installed, but you'll need to load it, preferably at the beginning of your report:

```
library(knitr)
dandruff %>% group_by(Treatment) %>%
summarise(n=n(), mean_flaking=mean(Flaking), sd_flaking=sd(Flaking)) -> summary
kable(summary)
```

Treatment	n	mean_flaking	sd_flaking
Keto	106	16.02830	0.9305149
Placebo	28	29.39286	1.5948827
PyrI	112	17.39286	1.1418110
PyrII	109	17.20183	1.3524999

End of aside.

Before I got distracted, we were talking about whether the distribution of PyrI was normal enough, given the sample size. Another way of thinking about this is to look at the bootstrapped sampling distribution of the sample mean for this group. I set the random number seed so that the results will be the same even if I run this again:

```
set.seed(457299)
```

```
dandruff %>% filter(Treatment == "PyrI") -> pyri
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(pyri$Flaking, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```

Oh yes, no problem with the normality there. (The discreteness of the population implies that each sample mean is some number of one-hundred-and-twelfths, so that the sampling distribution is also discrete, just less discrete than the data distribution. This is the reason for the little stair-steps in the plot.) In addition, the fact that the least normal distribution is normal enough means that the other distributions must also be OK. If you wanted to be careful, you would assess the smallest Placebo group as well, though that if anything is short-tailed and so would not be a problem anyway.

The other question is whether those spreads are equal enough. The easiest way is to look back at your summary table (that I reproduced above), cast your eye down the SD column, and make a call about whether they are equal enough. The large sample sizes *don't* help here, although see the end of the question for more discussion. I would call these "not grossly unequal" and call standard ANOVA good, but you are also entitled to call them different enough, and then you need to say that in your opinion we should have done a Welch ANOVA. Or, if you got your normal quantile plots before you did your ANOVA, you could actually *do* a Welch ANOVA.

I am not a fan of doing one test to see whether you can do another test,[15] but if you really want to, you can use something like Levene's test to test the null hypothesis that all the groups

[15]Because the true alpha for the combined procedure in which the test you do second depends on the result of the first test is no longer 0.05; you need to think about what that true alpha is. It might not be too bad here, because regular ANOVA and Welch ANOVA tend to come out similar unless the sample variances are very different, in the same way that the Welch and pooled two-sample tests do. But it is not something to take for granted.

have the same variance.[16] Levene's test lives in the package `car` that you might have to install first:

```
library(car)
leveneTest(Flaking~Treatment, data = dandruff)
```

```
Warning in leveneTest.default(y = y, group = group, ...): group coerced to
factor.
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value    Pr(>F)
group   3  6.2001 0.0004096 ***
      351
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Equal variances are resoundingly *rejected* here; the samples here have variances that are less equal than they would be if the populations all had the same variances. But that is really asking the wrong question: the one that matters is "does the inequality of variances that we saw here *matter* when it comes to doing the ANOVA?". With samples as big as we had, the variances could be declared unequal even if they were actually quite similar. This is another (different) angle on statistical significance (rather similar variances can be significantly different with large samples) vs. practical importance (does the fact that our sample variances are as different as they are matter to the ANOVA?). I do the Welch ANOVA in Extra 6, and you will see there whether it comes out much different than the regular ANOVA. See also Extra 7.

If your normal quantile plots looked like this:

```
ggplot(dandruff, aes(sample=Flaking)) + stat_qq() + stat_qq_line() +
facet_wrap(~Treatment)
```

[16]There are other tests you could use here. I like Levene's test because it works best when the samples are not normal, but the normality is OK here, so this is not so much of an issue. Of course, the time you want to be assessing equality of variances is when you have already seen sufficient normality, but that might have been because the samples were large rather than that they were especially normal in shape themselves.

with the *same* scales, you can use the slopes of the lines to judge equal spreads: either equal enough, or the `Placebo` line is a bit steeper than the others. If you did `scales = "free"`, you *cannot* do this, because you have essentially standardized your data before making the normal quantile plots.

It is *hugely* important to distinguish the null hypothesis (all the means are the same) from the assumptions behind the test (how you know that the P-value obtained from testing your null hypothesis can be trusted). These are separate things, and getting them straight is a vital part of being a good statistician. You might say that this is part of somebody else knowing how they, as someone hiring a statistician, can trust *you*.

Extra 5: Several ways to say what you conclude from the ANOVA:

- The null hypothesis, which says that all the shampoos have the same mean amount of flaking, is rejected. (Or say it in two sentences: what the null hypothesis is, and then what you're doing with it.)
- Not all the shampoos have the same mean amount of flaking.
- There are shampoos that differ in mean amount of flaking.

Some wrong or incomplete ways to say it:

- We reject the null hypothesis. (Meaning what, about the data?)
- we reject the null hypothesis that the means are different. (You have confused the null with the conclusion, and come out with something that is backwards.)

- the mean flaking for the treatments is different (this says that they are *all* different, but you don't know that yet.)

Extra 6: You might have been wondering how Welch's ANOVA would have played out, given that the placebo group measurements looked more variable than the others. Wonder no more:

```
oneway.test(Flaking~Treatment, data=dandruff)
```

```
	One-way analysis of means (not assuming equal variances)

data:  Flaking and Treatment
F = 595.03, num df = 3.00, denom df = 105.91, p-value < 2.2e-16
```

```
gamesHowellTest(Flaking~factor(Treatment), data=dandruff)
```

```
	Pairwise comparisons using Games-Howell test

data: Flaking by factor(Treatment)

        Keto    Placebo PyrI
Placebo 9.2e-14 -       -
PyrI    8.7e-14 < 2e-16 -
PyrII   2.1e-11 < 2e-16 0.67


P value adjustment method: none

alternative hypothesis: two.sided
```

The results are almost exactly the same: the conclusions are identical, and the P-values are even pretty much the same. The place where it would make a difference is when you are close to the boundary between rejecting and not. Here, our Tukey and Games-Howell P-values were all either close to 0 or about 0.6, whichever way we did it. So it didn't matter which one we did; you could justify using either. The regular ANOVA might have been a better choice for your report, though, because this is something your audience could reasonably be expected to

have heard of. The Welch ANOVA deserves to be as well-used as the Welch two-sample t-test, but it doesn't often appear in Statistics courses. (This course is an exception, of course!)

Extra 7: the general principle when you are not sure of the choice between two tests is to run them both. If the conclusions agree, as they do here, then it doesn't matter which one you run. If they disagree, then it matters, and you need to think more carefully about which test is the more appropriate one. (Usually, this is the test with the fewer assumptions, but not always.)

Another way to go is to do a simulation (of the ordinary ANOVA). Generate some data that are like what you actually have, and then in your simulation see whether your α is near to 0.05. Since we are talking about α here, the simulated data needs to have the same *mean* in every group, so that the null hypothesis is true, but SDs and sample sizes like the ones in the data (and of a normal shape). Let me build up the process. Let's start by making a dataframe that contains the sample sizes, means and SDs for the data we want to generate. The treatment names don't matter:

```
sim_from <- tribble(
~trt, ~n, ~mean, ~sd,
"A", 106, 0, 0.93,
"B", 28, 0, 1.59,
"C", 112, 0, 1.14,
"D", 109, 0, 1.35
)
sim_from
```

```
# A tibble: 4 x 4
  trt       n  mean    sd
  <chr> <dbl> <dbl> <dbl>
1 A       106     0  0.93
2 B        28     0  1.59
3 C       112     0  1.14
4 D       109     0  1.35
```

Starting from here, we want to set up drawing a lot of random samples from a normal distribution with the mean and SD shown, and the sample size shown. The way I like to do it[17] is to set up a list-column called `sim` that will index the simulations. I'm going to pretend I'm doing just three simulations, while I get my head around this, and then up it later after I have things working:

[17]At this moment, subject to change, etc etc.

```r
sim_from %>%
  mutate(sim = list(1:3))
```

```
# A tibble: 4 x 5
  trt       n  mean    sd sim
  <chr> <dbl> <dbl> <dbl> <list>
1 A       106     0  0.93 <int [3]>
2 B        28     0  1.59 <int [3]>
3 C       112     0  1.14 <int [3]>
4 D       109     0  1.35 <int [3]>
```

Then I `unnest` `sim` so that I have a place to draw each sample:

```r
sim_from %>%
  mutate(sim = list(1:3)) %>%
  unnest(sim)
```

```
# A tibble: 12 x 5
   trt       n  mean    sd   sim
   <chr> <dbl> <dbl> <dbl> <int>
 1 A       106     0  0.93     1
 2 A       106     0  0.93     2
 3 A       106     0  0.93     3
 4 B        28     0  1.59     1
 5 B        28     0  1.59     2
 6 B        28     0  1.59     3
 7 C       112     0  1.14     1
 8 C       112     0  1.14     2
 9 C       112     0  1.14     3
10 D       109     0  1.35     1
11 D       109     0  1.35     2
12 D       109     0  1.35     3
```

Now, working rowwise, I can draw a random sample from each normal distribution. The inputs to `rnorm` are the sample size, the mean, and the SD, in that order.[18]

[18] Or you can remember or look up the names to the inputs, and then you can have them in any order you like. But I know which order they come, which is good enough for me.

```r
sim_from %>%
  mutate(sim = list(1:3)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(y = list(rnorm(n, mean, sd)))
```

```
# A tibble: 12 x 6
# Rowwise:
   trt       n  mean    sd   sim y
   <chr> <dbl> <dbl> <dbl> <int> <list>
 1 A       106     0  0.93     1 <dbl [106]>
 2 A       106     0  0.93     2 <dbl [106]>
 3 A       106     0  0.93     3 <dbl [106]>
 4 B        28     0  1.59     1 <dbl [28]>
 5 B        28     0  1.59     2 <dbl [28]>
 6 B        28     0  1.59     3 <dbl [28]>
 7 C       112     0  1.14     1 <dbl [112]>
 8 C       112     0  1.14     2 <dbl [112]>
 9 C       112     0  1.14     3 <dbl [112]>
10 D       109     0  1.35     1 <dbl [109]>
11 D       109     0  1.35     2 <dbl [109]>
12 D       109     0  1.35     3 <dbl [109]>
```

This is looking similar in procedure to a bootstrap sampling distribution. If we were doing *that*, we would now make a new column containing something like the mean of each of those samples (and then make a picture of what we had). But this is different: we want to combine all of the random samples for each of the four treatments for one of the simulations, run an ANOVA on it, and get hold of the P-value. So we need to `unnest` those samples, and then combine them together properly. That goes something like this:

```r
sim_from %>%
  mutate(sim = list(1:3)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(y = list(rnorm(n, mean, sd))) %>%
  unnest(y) %>%
  nest_by(sim)
```

```
# A tibble: 3 x 2
# Rowwise:  sim
```

```
    sim               data
  <int> <list<tibble[,5]>>
1     1          [355 x 5]
2     2          [355 x 5]
3     3          [355 x 5]
```

What this has done is to create three mini-dataframes in the list-column `data` that have our generated random `y` and a column called `treatment`. What we want to do is to run the ordinary ANOVA on each of those dataframes in `data`, and, in a minute, get hold of the P-value. I think I need another `rowwise` first, because I want to work with the rows of the *new* dataframe:

```
sim_from %>%
  mutate(sim = list(1:3)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(y = list(rnorm(n, mean, sd))) %>%
  unnest(y) %>%
  nest_by(sim) %>%
  rowwise() %>%
  mutate(my_anova = list(aov(y ~ trt, data = data)))
```

```
# A tibble: 3 x 3
# Rowwise:
    sim               data my_anova
  <int> <list<tibble[,5]>> <list>
1     1          [355 x 5] <aov>
2     2          [355 x 5] <aov>
3     3          [355 x 5] <aov>
```

I know the P-value is in there somewhere, but I can't remember how to get hold of it. The easiest way is to load `broom` and pass the models into `tidy`, then take a look at that:

```
library(broom)
```

```
sim_from %>%
  mutate(sim = list(1:3)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(y = list(rnorm(n, mean, sd))) %>%
  unnest(y) %>%
```

```
  nest_by(sim) %>%
  rowwise() %>%
  mutate(my_anova = list(aov(y ~ trt, data = data))) %>%
  mutate(my_tidy = list(tidy(my_anova))) %>%
  unnest(my_tidy)
```

```
# A tibble: 6 x 9
    sim                data my_anova term      df  sumsq meansq statistic p.value
  <int> <list<tibble[,5]>> <list>   <chr>  <dbl>  <dbl>  <dbl>     <dbl>   <dbl>
1     1           [355 x 5] <aov>    trt        3  13.1    4.38     3.19   0.0239
2     1           [355 x 5] <aov>    Resid~   351 482.     1.37    NA     NA
3     2           [355 x 5] <aov>    trt        3   3.23   1.08     0.895  0.444
4     2           [355 x 5] <aov>    Resid~   351 422.     1.20    NA     NA
5     3           [355 x 5] <aov>    trt        3   5.35   1.78     1.19   0.313
6     3           [355 x 5] <aov>    Resid~   351 525.     1.50    NA     NA
```

Almost there. The rows that have P-values in them are the ones that have `trt` (the explanatory variable) in the `term`, so:

```
sim_from %>%
  mutate(sim = list(1:3)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(y = list(rnorm(n, mean, sd))) %>%
  unnest(y) %>%
  nest_by(sim) %>%
  rowwise() %>%
  mutate(my_anova = list(aov(y ~ trt, data = data))) %>%
  mutate(my_tidy = list(tidy(my_anova))) %>%
  unnest(my_tidy) %>%
  filter(term == "trt") %>%
  select(sim, p.value)
```

```
# A tibble: 3 x 2
    sim p.value
  <int>   <dbl>
1     1   0.817
2     2   0.273
3     3   0.169
```

Three simulated samples, each time one from each of the four treatments, and three P-values. So this works, and the remaining thing is to change the number of simulations from 3 to 1000 and run it again, saving the result:

```
sim_from %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(y = list(rnorm(n, mean, sd))) %>%
  unnest(y) %>%
  nest_by(sim) %>%
  rowwise() %>%
  mutate(my_anova = list(aov(y ~ trt, data = data))) %>%
  mutate(my_tidy = list(tidy(my_anova))) %>%
  unnest(my_tidy) %>%
  filter(term == "trt") %>%
  select(sim, p.value) -> sim_pval
sim_pval
```

```
# A tibble: 1,000 x 2
     sim p.value
   <int>   <dbl>
 1     1  0.687
 2     2  0.153
 3     3  0.508
 4     4  0.606
 5     5  0.665
 6     6  0.905
 7     7  0.854
 8     8  0.0548
 9     9  0.220
10    10  0.0301
# i 990 more rows
```

Now, the reason we were doing this was to see whether regular ANOVA worked properly on data from populations with different SDs. We know that the null hypothesis is true here (because all the true treatment means were equal to 0), so the probability of making a type I error by rejecting the null (that all the means are the same) should be 0.05. How close is it?

```
sim_pval %>%
  count(p.value <= 0.05)
```

```
# A tibble: 2 x 2
  `p.value <= 0.05`     n
  <lgl>             <int>
1 FALSE               909
2 TRUE                 91
```

91/1000 = 0.091. We are *too likely* to falsely reject the null. The regular ANOVA does not behave properly for data like ours.

That looks rather high. Is the proportion of times I am rejecting significantly different from 0.05? Testing null hypotheses about (single) proportions is done using `prop.test`. This uses the normal approximation to the binomial, with continuity correction:

```
prop.test(91, 1000, p = 0.05)
```

```
    1-sample proportions test with continuity correction

data:  91 out of 1000, null probability 0.05
X-squared = 34.532, df = 1, p-value = 4.194e-09
alternative hypothesis: true p is not equal to 0.05
95 percent confidence interval:
 0.07425035 0.11096724
sample estimates:
    p
0.091
```

Ah, now, that's interesting. A supposed $\alpha = 0.05$ test is actually rejecting around 9% of the time, which is significantly different from 0.05. This surprises me. So the ANOVA is actually *not* all that accurate.[19]

So now let's do the same simulation for the Welch ANOVA to see whether it's better:

```
sim_from %>%
  mutate(sim = list(1:1000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(y = list(rnorm(n, mean, sd))) %>%
  unnest(y) %>%
```

[19]When you get a simulation result that is not what you were expecting, there are two options to explore: either it really is different from your expectation, or there is something wrong with your code. I think my code is OK here, but do let me know if you see a problem with it.

```
    nest_by(sim) %>%
    rowwise() %>%
    mutate(my_anova = list(oneway.test(y ~ trt, data = data))) %>%
    mutate(pval = my_anova$p.value) -> sim_pval2
  sim_pval2
```

```
# A tibble: 1,000 x 4
# Rowwise:
     sim                data my_anova  pval
   <int> <list<tibble[,5]>> <list>   <dbl>
 1     1           [355 x 5] <htest>  0.748
 2     2           [355 x 5] <htest>  0.777
 3     3           [355 x 5] <htest>  0.352
 4     4           [355 x 5] <htest>  0.629
 5     5           [355 x 5] <htest>  0.341
 6     6           [355 x 5] <htest>  0.203
 7     7           [355 x 5] <htest>  0.205
 8     8           [355 x 5] <htest>  0.583
 9     9           [355 x 5] <htest>  0.273
10    10           [355 x 5] <htest>  0.638
# i 990 more rows
```

This one is a bit easier because `oneway.test` has a thing called `p.value` that you can just pull out. No need to use `tidy` here.

How many of *those* P-values are less than 0.05?

```
  sim_pval2 %>%
    count(pval <= 0.05)
```

```
# A tibble: 2 x 2
# Rowwise:
  `pval <= 0.05`     n
  <lgl>          <int>
1 FALSE            951
2 TRUE              49
```

That couldn't be much closer to the mark:

```
  prop.test(49, 1000, p = 0.05)
```

```
    1-sample proportions test with continuity correction

data:  49 out of 1000, null probability 0.05
X-squared = 0.0052632, df = 1, p-value = 0.9422
alternative hypothesis: true p is not equal to 0.05
95 percent confidence interval:
 0.03682698 0.06475244
sample estimates:
    p
0.049
```

This one is on the money.[20] The proportion of times our simulation falsely rejects is not significantly different from 0.05. So this investigation says that the Welch ANOVA is much more trustworthy for data resembling what we observed.

■

[20]You could redo this with 10,000 simulations to convince yourself further.

15 Tidying data

```
library(tidyverse)
```

15.1 Baseball and softball spaghetti

On a previous assignment, we found that students could throw a baseball further than they could throw a softball. In this question, we will make a graph called a "spaghetti plot" to illustrate this graphically. (The issue previously was that the data were matched pairs: the same students threw both balls.)

This seems to work most naturally by building a pipe, a line or two at a time. See if you can do it that way. (If you can't make it work, use lots of temporary data frames, one to hold the result of each part.)

(a) Read in the data again from link. The variables had no names, so supply some, as you did before.

(b) Create a new column that is the students turned into a `factor`, adding it to your data frame. The reason for this will become clear later.

(c) Collect together all the throwing distances into one column, making a second column that says which ball was thrown.

(d) Using your new data frame, make a "scatterplot" of throwing distance against type of ball.

(e) Add two things to your plot: something that will distinguish the students by colour (this works best if the thing distinguished by colour is a factor),[1] and something that will join the two points for the same student by a line.

(f) The legend is not very informative. Remove it from the plot, using `guides`.

(g) What do you see on the final spaghetti plot? What does that tell you about the relative distances a student can throw a baseball vs. a softball? Explain briefly, blah blah blah.

[1] You can try it without. See below.

15.2 Ethanol and sleep time in rats

A biologist wished to study the effects of ethanol on sleep time in rats. A sample of 20 rats (all the same age) was selected, and each rat was given an injection having a particular concentration (0, 1, 2 or 4 grams per kilogram of body weight) of ethanol. These are labelled `e0, e1, e2, e4`. The "0" treatment was a control group. The rapid eye movement (REM) sleep time was then recorded for each rat. The data are in link.

(a) Read the data in from the file. Check that you have four rows of observations and five columns of sleep times.

(b) Unfortunately, the data are in the wrong format. All the sleep times for each treatment group are on one row, and we should have *one* column containing *all* the sleep times, and the corresponding row should show which treatment group that sleep time came from. Transform this data frame into one that you could use for modelling or making graphs.

(c) Using your new data frame, make side-by-side boxplots of sleep time by treatment group.

(d) In your boxplots, how does the median sleep time appear to depend on treatment group?

(e) There is an assumption about spread that the analysis of variance needs in order to be reliable. Do your boxplots indicate that this assumption is satisfied for these data, bearing in mind that you have only five observations per group?

(f) Run an analysis of variance to see whether sleep time differs significantly among treatment groups. What do you conclude?

(g) Would it be a good idea to run Tukey's method here? Explain briefly why or why not, and if you think it would be a good idea, run it.

(h) What do you conclude from Tukey's method? (This is liable to be a bit complicated.) Is there a treatment that is clearly best, in terms of the sleep time being largest?

15.3 Growth of tomatoes

A biology graduate student exposed each of 32 tomato plants to one of four different colours of light (8 plants to each colour). The growth rate of each plant, in millimetres per week, was recorded. The data are in link.

(a) Read the data into R and confirm that you have 8 rows and 5 columns of data.

(b) Re-arrange the data so that you have *one* column containing all the growth rates, and another column saying which colour light each plant was exposed to. (The aim here is to produce something suitable for feeding into `aov` later.)

(c) Save the data in the new format to a text file. This is most easily done using `write_csv`, which is the opposite of `read_csv`. It requires two things: a data frame, and the name of a file to save in, which should have a `.csv` extension.

(d) Make a suitable boxplot, and use it to assess the assumptions for ANOVA. What do you conclude? Explain briefly.

(e) Run (regular) ANOVA on these data. What do you conclude? (Optional extra: if you think that some other variant of ANOVA would be better, run that as well and compare the results.)

(f) If warranted, run a suitable follow-up. (If not warranted, explain briefly why not.)

15.4 Pain relief in migraine headaches (again)

The data in link are from a study of pain relief in migraine headaches. Specifically, 27 subjects were randomly assigned to receive *one* of three pain relieving drugs, labelled A, B and C. Each subject reported the number of hours of pain relief they obtained (that is, the number of hours between taking the drug and the migraine symptoms returning). A higher value is therefore better. Can we make some recommendation about which drug is best for the population of migraine sufferers?

(a) Read in and display the data. Take a look at the data file first, and see if you can say why `read_table` will work and `read_delim` will not.

(b) What is it about the experimental design that makes a one-way analysis of variance plausible for data like this?

(c) What is wrong with the current format of the data as far as doing a one-way ANOVA analysis is concerned? (This is related to the idea of whether or not the data are "tidy".)

(d) "Tidy" the data to produce a data frame suitable for your analysis.

(e) Go ahead and run your one-way ANOVA (and Tukey if necessary). Assume for this that the pain relief hours in each group are sufficiently close to normally distributed with sufficiently equal spreads.

(f) What recommendation would you make about the best drug or drugs? Explain briefly.

15.5 Location, species and disease in plants

The table below is a "contingency table", showing frequencies of diseased and undiseased plants of two different species in two different locations:

```
Species     Disease present         Disease absent
Location X Location Y  Location X Location Y
A            44         12          38        10
B            28         22          20        18
```

The data were saved as link. In that file, the columns are coded by two letters: a `p` or an `a` to denote presence or absence of disease, and an `x` or a `y` to denote location X or Y. The data are separated by multiple spaces and aligned with the variable names.

(a) Read in and display the data.

(b) Explain briefly how these data are not "tidy".

(c) Use a suitable `tidyr` tool to get all the things that are the same into a single column. (You'll need to make up a temporary name for the other new column that you create.) Show your result.

(d) Explain briefly how the data frame you just created is still not "tidy" yet.

(e) Use one more `tidyr` tool to make these data tidy, and show your result.

(f) Let's see if we can re-construct the original contingency table (or something equivalent to it). Use the function `xtabs`. This requires first a model formula with the frequency variable on the left of the squiggle, and the other variables separated by plus signs on the right. Second it requires a data frame, with `data=`. Feed your data frame from the previous part into `xtabs`. Save the result in a variable and display the result.

(g) Take the output from the last part and feed it into the function `ftable`. How has the output been changed? Which do you like better? Explain briefly.

15.6 Mating songs in crickets

Male tree crickets produce "mating songs" by rubbing their wings together to produce a chirping sound. It is hypothesized that female tree crickets identify males of the correct species by how fast (in chirps per second) the male's mating song is. This is called the "pulse rate". Some data for two species of crickets are in link. The columns, which are unlabelled, are temperature and pulse rate (respectively) for *Oecanthus exclamationis* (first two columns) and *Oecanthus niveus* (third and fourth columns). The columns are separated by tabs. There are

some missing values in the first two columns because fewer *exclamationis* crickets than *niveus* crickets were measured. The research question is whether males of the different species have different average pulse rates. It is also of interest to see whether temperature has an effect, and if so, what. Before we get to that, however, we have some data organization to do.

(a) Read in the data, allowing for the fact that you have no column names. You'll see that the columns have names `X1` through `X4`. This is OK.

(b) Tidy these untidy data, going as directly as you can to something tidy. (Some later parts show you how it used to be done.) Begin by: (i) adding a column of row numbers, (ii) `rename`-ing the columns to species name, an underscore, and the variable contents (keeping `pulserate` as one word), and then use `pivot_longer`. Note that the column names encode *two* things.

(c) If you found (b) a bit much to take in, the rest of the way we take a rather more leisurely approach towards the tidying.

These data are rather far from being tidy. There need to be three variables, temperature, pulse rate and species, and there are $14 + 17 = 31$ observations altogether. This one is tricky in that there are temperature and pulse rate for each of two levels of a factor, so I'll suggest combining the temperature and chirp rate together into one thing for each species, then pivoting them longer ("combining"), then pivoting them wider again ("splitting"). Create new columns, named for each species, that contain the temperature and pulse rate for that species in that order, `unite`d together. For the rest of this question, start from the data frame you read in, and build a pipe, one or two steps at a time, to save creating a lot of temporary data frames.

(d) The two columns `exclamationis` and `niveus` that you just created are both temperature-pulse rate combos, but for different species. Collect them together into one column, labelled by species. (This is a straight `tidyr pivot_longer`, even though the columns contain something odd-looking.)

(e) Now split up the temperature-pulse combos at the underscore, into two separate columns. This is `separate`. When specifying what to separate by, you can use a number ("split after this many characters") or a piece of text, in quotes ("when you see this text, split at it").

(f) Almost there. Temperature and pulse rate are still text (because `unite` turned them into text), but they should be numbers. Create new variables that are numerical versions of temperature and pulse rate (using `as.numeric`). Check that you have no extraneous variables (and, if necessary, get rid of the ones you don't want). (Species is also text and really ought to be a factor, but having it as text doesn't seem to cause any problems.) You can, if you like, use `parse_number` instead of `as.numeric`. They should both work. The distinction I prefer to make is that `parse_number` is good for text with a number in

it (that we want to pull the number out of), while `as.numeric` is for turning something that looks like a number but isn't one into a genuine number.[2]

15.7 Number 1 songs

The data file link contains a lot of information about songs popular in 2000. This dataset is untidy. Our ultimate aim is to answer "which song occupied the #1 position for the largest number of weeks?". To do that, we will build a pipe that starts from the data frame read in from the URL above, and finishes with an answer to the question. I will take you through this step by step. Each part will involve adding something to the pipe you built previously (possibly after removing a line or two that you used to display the previous result).

(a) Read the data and display what you have.

(b) The columns `x1st.week` through `x76th.week` contain the rank of each song in the Billboard chart in that week, with week 1 being the first week that the song appeared in the chart. Convert all these columns into two: an indication of week, called `week`, and of rank, called `rank`. Most songs appeared in the Billboard chart for a lot less than 76 weeks, so there are missing values, which you want to remove. (I say "indication of week" since this will probably be text at the moment). Display your new data frame. Do you have fewer columns? Why do you have a lot more rows? Explain briefly.

(c) Both your `week` and `rank` columns are (probably) text. Create new columns that contain just the numeric values, and display just your new columns, again adding onto the end of your pipe. If it so happens that `rank` is already a number, leave it as it is.

(d) The meaning of your week-number column is that it refers to the number of weeks *after* the song first appeared in the Billboard chart. That is, if a song's first appearance (in `date.entered)` is July 24, then week 1 is July 24, week 2 is July 31, week 3 is August 7, and so on. Create a column `current` by adding the appropriate number of *days*, based on your week number, to `date.entered`. Display `date.entered`, your week number, and `current` to show that you have calculated the right thing. Note that you can add a number of days onto a date and you will get another date.

(e) Reaching the #1 rank on the Billboard chart is one of the highest accolades in the popular music world. List all the songs that reached `rank` 1. For these songs, list the artist (as given in the data set), the song title, and the date(s) for which the song was ranked number 1. Arrange the songs in date order of being ranked #1. Display all the songs (I found 55 of them).

[2] You could just as well make the point that the text 20.8 contains the number 20.8 and nothing else, so that parsing it as text in search of a number will pull out 20.8 as a number. If that logic works for you, go with it.

(f) Use R to find out which song held the #1 rank for the largest number of weeks. For this, you can assume that the song titles are all unique (if it's the same song title, it's the same song), but the artists might not be (for example, Madonna might have had two different songs reach the #1 rank). The information you need is in the output you obtained for the previous part, so it's a matter of adding some code to the end of that. The last mark was for displaying *only* the song that was ranked #1 for the largest number of weeks, or for otherwise making it easy to see which song it was.

15.8 Bikes on College

The City of Toronto collects all kinds of data on aspects of life in the city. See link. One collection of data is records of the number of cyclists on certain downtown streets. The data in link are a record of the cyclists on College Street on the block west from Huron to Spadina on September 24, 2010. In the spreadsheet, each row relates to one cyclist. The first column is the time the cyclist was observed (to the nearest 15 minutes). After that, there are four pairs of columns. The observer filled in (exactly) one X in each pair of columns, according to whether (i) the cyclist was male or female, (ii) was or was not wearing a helmet, (iii) was or was not carrying a passenger on the bike, (iv) was or was not riding on the sidewalk. We want to create a tidy data frame that has the time in each row, and has columns containing appropriate values, often `TRUE` or `FALSE`, for each of the four variables measured.

I will lead you through the process, which will involve developing a (long) pipe, one step at a time.

(a) Take a look at the spreadsheet (using Excel or similar: this may open when you click the link). Are there any obvious header rows? Is there any extra material before the data start? Explain briefly.

(b) Read the data into an R data frame. Read *without* headers, and instruct R how many lines to skip over using `skip=` and a suitable number. When this is working, display the first few lines of your data frame. Note that your columns have names `X1` through `X9`.

(c) What do you notice about the times in your first column? What do you think those "missing" times should be?

(d) Find something from the `tidyverse` that will fill[3] in those missing values with the right thing. Start a pipe from the data frame you read in, that updates the appropriate column with the filled-in times.

(e) R's `ifelse` function works like `=IF` in Excel. You use it to create values for a new variable, for example in a `mutate`. The first input to it is a logical condition (something that is either true or false); the second is the value your new variable should take if the

[3] Oh, what a giveaway.

condition is true, and the third is the value of your new variable if the condition is false. Create a new column `gender` in your data frame that is "male" or "female" depending on the value of your `X2` column, using `mutate`. (You can assume that exactly one of the second and third columns has an `X` in it.) Add your code to the end of your pipe and display (the first 10 rows of) the result.

(f) Create variables `helmet`, `passenger` and `sidewalk` in your data frame that are `TRUE` if the "Yes" column contains `X` and `FALSE` otherwise. This will use `mutate` again, but you don't need `ifelse`: just set the variable equal to the appropriate logical condition. As before, the best way to create these variables is to test the appropriate things for missingness. Note that you can create as many new variables as you like in one `mutate`. Show the first few lines of your new data frame. (Add your code onto the end of the pipe you made above.)

(g) Finally (for the data manipulation), get rid of all the original columns, keeping only the new ones that you created. Save the results in a data frame and display its first few rows.

(h) The next few parts are a quick-fire analysis of the data set. They can all be solved using `count`. How many male and how many female cyclists were observed in total?

(i) How many male and female cyclists were not wearing helmets?

(j) How many cyclists were riding on the sidewalk *and* carrying a passenger?

(k) What was the busiest 15-minute period of the day, and how many cyclists were there then?

15.9 Feeling the heat

In summer, the city of Toronto issues Heat Alerts for "high heat or humidity that is expected to last two or more days". The precise definitions are shown at link. During a heat alert, the city opens Cooling Centres and may extend the hours of operation of city swimming pools, among other things. All the heat alert days from 2001 to 2016 are listed at link.

The word "warning" is sometimes used in place of "alert" in these data. They mean the same thing.[4]

(a) Read the data into R, and display the data frame. Note that there are four columns:

- a numerical `id` (numbered upwards from the first Heat Alert in 2001; some of the numbers are missing)
- the `date` of the heat alert, in year-month-day format with 4-digit years.

[4]Unlike *thunderstorm watch* and *thunderstorm warning*, which mean different things.

- a text `code` for the type of heat alert
- `text` describing the kind of heat alert. This can be quite long.

(b) In your data frame, are the dates stored as genuine dates or as text? How can you tell?

(c) Which different heat alert codes do you have, and how many of each?

(d) Use the `text` in your dataset (or look back at the original data file) to describe briefly in your own words what the various codes represent.

(e) How many (regular and extended) heat alert events are there altogether? A heat alert event is a stretch of consecutive days, on all of which there is a heat alert or extended heat alert. Hints: (i) you can answer this from output you already have; (ii) how can you tell when a heat alert event *starts*?

(f) We are going to investigate how many heat alert days there were in each year. To do that, we have to extract the year from each of our dates.

(g) Count the number of heat alert days for each year, by tabulating the year variable. Looking at this table, would you say that there have been more heat alert days in recent years? Explain (very) briefly.

15.10 Isoflavones

The plant called kudzu was imported to the US South from Japan. It is rich in isoflavones, which are believed to be beneficial for bones. In a study, rats were randomly assigned to one of three diets: one with a low dose of isoflavones from kudzu, one with a high dose, and a control diet with no extra isoflavones. At the end of the study, each rat's bone density was measured, in milligrams per square centimetre. The data as recorded are shown in http://ritsokiguess.site/isoflavones.txt.[5] There are 15 observations for each treatment, and hence 45 altogether.

Here are some code ideas you might need to use later, all part of the `tidyverse`. You may need to find out how they work.

- `col_names` (in the `read_` functions)
- `convert` (in various `tidyverse` functions)
- `fill`
- `na_if`

[5]Evidently the units were chosen for ease of recording; had the values been in grams instead, the person recording the data would have had to put a 0 and a decimal point on the front of each value. This is the old meaning of the word "coding"; making the data values be whole numbers and/or small deviations from something makes them easier to record, and in pre-computer days easier to calculate with. You will also see the same word used for classifying survey responses into categories, which is similar but not quite the same thing.

- `rename`
- `separate_rows`
- `skip` (in the `read_` functions)
- `values_drop_na` (in the `pivot_` functions)

If you use any of these, *cite* the webpage(s) or other source(s) where you learned about them.

(a) Take a look at the data file. Describe briefly what you see.

(b) Read in the data, using `read_table`, and get it into a tidy form, suitable for making a graph. This means finishing with (at least) a column of treatments with a suitable name (the treatments will be text) and a column of bone density values (numbers), one for each rat. You can have other columns as well; there is no obligation to get rid of them. Describe your process clearly enough that someone new to this data set would be able to understand what you have done and reproduce it on another similar dataset. Before you begin, think about whether or not you want to keep the column headers that are in the data file or not. (It can be done either way, but one way is easier than the other.)

(c) The statistician on this study is thinking about running an ordinary analysis of variance to compare the bone mineral density for the different treatments. Obtain a plot from your tidy dataframe that will help her decide whether that is a good idea.

(d) Based on your graph, and any additional graphs you wish to draw, what analysis would you recommend for this dataset? Explain briefly. (Don't do the analysis.)

15.11 Jocko's Garage

Insurance adjusters are concerned that Jocko's Garage is giving estimates for repairing car damage that are too high. To see whether this is indeed the case, ten cars that had been in collisions were taken to both Jocko's Garage and another garage, and the two estimates for repair were recorded. The data as recorded are here.

(a) Take a look at the data file (eg. by using your web browser). How are the data laid out? Do there appear to be column headers?

(b) Read in and display the data file, bearing in mind what you just concluded about it. What names did the columns acquire?

(c) Make this data set tidy. That is, you need to end up with columns containing the repair cost estimates at each of the two garages and also identifying the cars, with each observation on one row. Describe your thought process. (It needs to be possible for the reader to follow your description and understand why it works.) Save your tidy dataframe.

(d) Make a suitable graph to assess the comparison of interest, and say briefly what your graph tells you.

(e) Carry out a test to make an appropriate comparison of the mean estimates. What do you conclude, in the context of the data?

15.12 Tidying electricity consumption

How does the consumption of electricity depend on temperature? To find out, a short-term study was carried out by a utility company based in a certain area. For a period of two years, the average monthly temperature was recorded (in degrees Fahrenheit), the mean daily demand for electricity per household (in kilowatt hours), and the cost per kilowatt hour of electricity for that year (8 cents for the first year and 10 cents for the second, which it will be easiest to treat as categorical).

The data were laid out in an odd way, as shown in http://ritsokiguess.site/datafiles/utils.txt, in aligned columns: the twelve months of temperature were laid out on *two* lines for the first year, then the twelve months of consumption for the first year on the next two lines, and then four more lines for the second year laid out the same way. Thus the temperature of 31 in the first line goes with the consumption of 55 in the *third* line, and the last measurements for that year are the 78 at the end of the second line (temperature) and 73 at the end of the fourth line (consumption). Lines 5 through 8 of the data file are the same thing for the second year (when electricity was more expensive).

The data seem to have been laid out in order of temperature, rather than in order of months, which I would have thought would make more sense. But this is what we have.

(a) Read in and display the data file, bearing in mind that it has *no column names*.

(b) Arrange these data tidily, so that there is a column of price (per kilowatt hour), a column of temperatures, and a column of consumptions. Describe your process, including why you got list-columns (if you did) and what you did about them (if necessary).

(c) Make a suitable graph of temperature, consumption and price in your tidy dataframe. Add smooth trends if appropriate. If you were unable to get the data tidy, use my tidy version here. (If you need the other file, right-click on "here" and Copy Link Address.)

(d) What patterns or trends do you see on your graph? Do they make practical sense? There are two things I would like you to comment on.

15.13 Tidy blood pressure

Going to the dentist is scary for a lot of people. One way in which this might show up is that people might have higher blood pressure on average before their dentist's appointment than an hour after the appointment is done. Ten randomly-chosen individuals have their (systolic[6]) blood pressure measured while they are in a dentist's waiting room, and then again one hour after their appointment is finished.

You might have seen a tidy version of this data set before.

The data as I originally received it is in http://ritsokiguess.site/datafiles/blood_pressure2.csv.

(a) Read in and display the data as originally received.

(b) Describe briefly how the data you read in is not tidy, bearing in mind how the data were collected and how they would be analysed.

(c) Produce a tidy dataframe from the one you read in from the file. (How many rows should you have?)

(d) What kind of test might you run on these data? Explain briefly.

(e) Draw a suitable graph of these data.

My solutions follow:

15.14 Baseball and softball spaghetti

On a previous assignment, we found that students could throw a baseball further than they could throw a softball. In this question, we will make a graph called a "spaghetti plot" to illustrate this graphically. (The issue previously was that the data were matched pairs: the same students threw both balls.)

This seems to work most naturally by building a pipe, a line or two at a time. See if you can do it that way. (If you can't make it work, use lots of temporary data frames, one to hold the result of each part.)

(a) Read in the data again from link. The variables had no names, so supply some, as you did before.

[6] A blood pressure is usually given as two numbers, like "120 over 80". The first number, which is the one shown in our data, is called the systolic blood pressure. It is the pressure in the arteries when the heart is pumping. The second is called the diastolic blood pressure, and it is the pressure in the arteries when the heart is resting.

Solution

Literal copy and paste:

```
myurl <- "http://ritsokiguess.site/datafiles/throw.txt"
throws <- read_delim(myurl, " ", col_names = c("student", "baseball", "softball"))
```

```
Rows: 24 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (3): student, baseball, softball

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
throws
```

```
# A tibble: 24 x 3
   student baseball softball
     <dbl>    <dbl>    <dbl>
 1       1       65       57
 2       2       90       58
 3       3       75       66
 4       4       73       61
 5       5       79       65
 6       6       68       56
 7       7       58       53
 8       8       41       41
 9       9       56       44
10      10       70       65
# i 14 more rows
```

■

(b) Create a new column that is the students turned into a `factor`, adding it to your data frame. The reason for this will become clear later.

Solution

Feed `student` into `factor`, creating a new column with `mutate`:

```
throws %>% mutate(fs = factor(student))
```

```
# A tibble: 24 x 4
   student baseball softball fs
     <dbl>    <dbl>    <dbl> <fct>
 1       1       65       57 1
 2       2       90       58 2
 3       3       75       66 3
 4       4       73       61 4
 5       5       79       65 5
 6       6       68       56 6
 7       7       58       53 7
 8       8       41       41 8
 9       9       56       44 9
10      10       70       65 10
# i 14 more rows
```

This doesn't look any different from the original student numbers, but note the variable type at the top of the column.

■

(c) Collect together all the throwing distances into one column, making a second column that says which ball was thrown.

Solution

Use `pivot_longer`. It goes like this:

```
throws %>%
  mutate(fs = factor(student)) %>%
  pivot_longer(baseball:softball, names_to="ball", values_to="distance")
```

```
# A tibble: 48 x 4
   student fs    ball     distance
     <dbl> <fct> <chr>       <dbl>
 1       1 1     baseball       65
 2       1 1     softball       57
 3       2 2     baseball       90
 4       2 2     softball       58
 5       3 3     baseball       75
 6       3 3     softball       66
 7       4 4     baseball       73
 8       4 4     softball       61
 9       5 5     baseball       79
```

```
10       5 5     softball       65
# i 38 more rows
```

The `names_to` is the name of a new categorical column whose values will be what is currently column names, and the `values_to` names a new quantitative (usually) column that will hold the values in those columns that you are making longer.

If you want to show off a little, you can use a select-helper, noting that the columns you want to make longer all end in "ball":

```
throws %>%
  mutate(fs = factor(student)) %>%
  pivot_longer(ends_with("ball"), names_to="ball", values_to="distance")
```

```
# A tibble: 48 x 4
   student fs    ball     distance
     <dbl> <fct> <chr>       <dbl>
 1       1 1     baseball       65
 2       1 1     softball       57
 3       2 2     baseball       90
 4       2 2     softball       58
 5       3 3     baseball       75
 6       3 3     softball       66
 7       4 4     baseball       73
 8       4 4     softball       61
 9       5 5     baseball       79
10       5 5     softball       65
# i 38 more rows
```

The same result. Use whichever you like.

■

(d) Using your new data frame, make a "scatterplot" of throwing distance against type of ball.

Solution

The obvious thing. No data frame in the `ggplot` because it's the data frame that came out of the previous part of the pipeline (that doesn't have a name):

```
throws %>%
  mutate(fs = factor(student)) %>%
```

```
pivot_longer(baseball:softball, names_to="ball", values_to="distance") %>%
ggplot(aes(x = ball, y = distance)) + geom_point()
```

This is an odd type of scatterplot because the x-axis is actually a categorical variable. It's really what would be called something like a dotplot. We'll be using this as raw material for the plot we actually want.

What this plot is missing is an indication of which student threw which ball. As it stands now, it could be an inferior version of a boxplot of distances thrown for each ball (which would imply that they are two independent sets of students, something that is not true).

■

(e) Add two things to your plot: something that will distinguish the students by colour (this works best if the thing distinguished by colour is a factor),[7] and something that will join the two points for the same student by a line.

Solution

A `colour` and a `group` in the `aes`, and a `geom_line`:

[7] You can try it without. See below.

```
throws %>%
  mutate(fs = factor(student)) %>%
  pivot_longer(baseball:softball, names_to="ball", values_to="distance") %>%
  ggplot(aes(x = ball, y = distance, group = fs, colour = fs)) +
  geom_point() + geom_line()
```

You can see what happens if you use the student as a number:

```
throws %>%
  mutate(fs = factor(student)) %>%
  pivot_longer(baseball:softball, names_to="ball", values_to="distance") %>%
  ggplot(aes(x = ball, y = distance, group = student, colour = student)) +
  geom_point() + geom_line()
```

Now the student numbers are distinguished as a shade of blue (on an implied continuous scale: even a nonsensical fractional student number like 17.5 would be a shade of blue). This is not actually so bad here, because all we are trying to do is to distinguish the students sufficiently from each other so that we can see where the spaghetti strands go. But I like the multi-coloured one better.

■

(f) The legend is not very informative. Remove it from the plot, using `guides`.

Solution

You may not have seen this before. Here's what to do: Find what's at the top of the legend that you want to remove. Here that is `fs`. Find where `fs` appears in your `aes`. It actually appears in two places: in `group` and `colour`. I think the legend we want to get rid of is actually the `colour` one, so we do this:

```
throws %>%
  mutate(fs = factor(student)) %>%
  pivot_longer(baseball:softball, names_to="ball", values_to="distance") %>%
  ggplot(aes(x = ball, y = distance, group = fs, colour = fs)) +
  geom_point() + geom_line() +
  guides(colour = F)
```

```
Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
of ggplot2 3.3.4.
```

That seems to have done it.

■

(g) What do you see on the final spaghetti plot? What does that tell you about the relative distances a student can throw a baseball vs. a softball? Explain briefly, blah blah blah.

Solution

Most of the spaghetti strands go downhill from baseball to softball, or at least very few of them go uphill. That tells us that most students can throw a baseball further than a softball. That was the same impression that the matched-pairs t-test gave us. But the spaghetti plot tells us something else. If you look carefully, you see that most of the big drops are for students who could throw a baseball a long way. These students also threw a softball further than the other students, but not by as much. Most of the spaghetti strands in the bottom half of the plot go more or less straight across. This indicates that students who cannot throw a baseball very far will throw a softball about the same distance as they threw the baseball. There is an argument you could make here that the difference between distances thrown is a *proportional* one, something like "a student typically throws a baseball 20% further than a softball". That could be assessed by comparing not the distances themselves, but the logs of the distances: in other words, making a log transformation of all the distances. (Distances have a lower limit

of zero, so you might expect observed distances to be skewed to the right, which is another argument for making some kind of transformation.)

■

15.15 Ethanol and sleep time in rats

A biologist wished to study the effects of ethanol on sleep time in rats. A sample of 20 rats (all the same age) was selected, and each rat was given an injection having a particular concentration (0, 1, 2 or 4 grams per kilogram of body weight) of ethanol. These are labelled `e0, e1, e2, e4`. The "0" treatment was a control group. The rapid eye movement (REM) sleep time was then recorded for each rat. The data are in link.

(a) Read the data in from the file. Check that you have four rows of observations and five columns of sleep times.

Solution

Separated by single spaces:

```
my_url <- "http://ritsokiguess.site/datafiles/ratsleep.txt"
sleep1 <- read_delim(my_url, " ")
```

```
Rows: 4 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): treatment
dbl (5): obs1, obs2, obs3, obs4, obs5

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
sleep1
```

```
# A tibble: 4 x 6
  treatment  obs1  obs2  obs3  obs4  obs5
  <chr>     <dbl> <dbl> <dbl> <dbl> <dbl>
1 e0         88.6  73.2  91.4  68    75.2
2 e1         63    53.9  69.2  50.1  71.5
3 e2         44.9  59.5  40.2  56.3  38.7
4 e4         31    39.6  45.3  25.2  22.7
```

There are six columns, but one of them labels the groups, and there are correctly five columns of sleep times.

I used a "temporary" name for my data frame, because I'm going to be doing some processing on it in a minute, and I want to reserve the name `sleep` for my processed data frame.

■

(b) Unfortunately, the data are in the wrong format. All the sleep times for each treatment group are on one row, and we should have *one* column containing *all* the sleep times, and the corresponding row should show which treatment group that sleep time came from. Transform this data frame into one that you could use for modelling or making graphs.

Solution

We will want *one* column of sleep times, with an additional categorical column saying what observation each sleep time was within its group (or, you might say, we don't really care about that much, but that's what we are going to get).

The columns `obs1` through `obs5` are different in that they are different observation numbers ("replicates", in the jargon). I'll call that `rep`. What makes them the same is that they are all sleep times. Columns `obs1` through `obs5` are the ones we want to combine, thus. Here is where I use the name `sleep`: I save the result of the `pivot_longer` into a data frame `sleep`. Note that I also used the brackets-around-the-outside to display what I had, so that I didn't have to do a separate display. This is a handy way of saving *and* displaying in one shot:

```
(sleep1 %>%
  pivot_longer(-treatment, names_to="rep", values_to="sleeptime") -> sleep)
```

```
# A tibble: 20 x 3
   treatment rep   sleeptime
   <chr>     <chr>     <dbl>
 1 e0        obs1       88.6
 2 e0        obs2       73.2
 3 e0        obs3       91.4
 4 e0        obs4       68
 5 e0        obs5       75.2
 6 e1        obs1       63
 7 e1        obs2       53.9
 8 e1        obs3       69.2
 9 e1        obs4       50.1
10 e1        obs5       71.5
11 e2        obs1       44.9
12 e2        obs2       59.5
```

```
13 e2        obs3       40.2
14 e2        obs4       56.3
15 e2        obs5       38.7
16 e4        obs1       31
17 e4        obs2       39.6
18 e4        obs3       45.3
19 e4        obs4       25.2
20 e4        obs5       22.7
```

Typically in this kind of work, you have a lot of columns that need to be made longer, and a much smaller number of columns that need to be repeated as necessary. You can either specify all the columns to make longer, or you can specify "not" the other columns. Above, my first input to `pivot_longer` was "everything but treatment", but you could also do it like this:

```
sleep1 %>%
  pivot_longer(obs1:obs5, names_to="rep", values_to="sleeptime")
```

```
# A tibble: 20 x 3
   treatment rep   sleeptime
   <chr>     <chr>     <dbl>
 1 e0        obs1       88.6
 2 e0        obs2       73.2
 3 e0        obs3       91.4
 4 e0        obs4       68
 5 e0        obs5       75.2
 6 e1        obs1       63
 7 e1        obs2       53.9
 8 e1        obs3       69.2
 9 e1        obs4       50.1
10 e1        obs5       71.5
11 e2        obs1       44.9
12 e2        obs2       59.5
13 e2        obs3       40.2
14 e2        obs4       56.3
15 e2        obs5       38.7
16 e4        obs1       31
17 e4        obs2       39.6
18 e4        obs3       45.3
19 e4        obs4       25.2
20 e4        obs5       22.7
```

or like this:

```r
sleep1 %>%
  pivot_longer(starts_with("obs"), names_to="rep", values_to="sleeptime")
```

```
# A tibble: 20 x 3
   treatment rep   sleeptime
   <chr>     <chr>     <dbl>
 1 e0        obs1       88.6
 2 e0        obs2       73.2
 3 e0        obs3       91.4
 4 e0        obs4       68
 5 e0        obs5       75.2
 6 e1        obs1       63
 7 e1        obs2       53.9
 8 e1        obs3       69.2
 9 e1        obs4       50.1
10 e1        obs5       71.5
11 e2        obs1       44.9
12 e2        obs2       59.5
13 e2        obs3       40.2
14 e2        obs4       56.3
15 e2        obs5       38.7
16 e4        obs1       31
17 e4        obs2       39.6
18 e4        obs3       45.3
19 e4        obs4       25.2
20 e4        obs5       22.7
```

This one was a little unusual in that usually with these you have the *treatments* in the columns and the replicates in the rows. It doesn't matter, though: `pivot_longer` handles both cases.

We have 20 rows of 3 columns. I got all the rows, but you will probably get an output with ten rows as usual, and will need to click Next to see the last ten rows. The initial display will say how many rows (20) and columns (3) you have.

The column `rep` is not very interesting: it just says which observation each one was within its group.[8] The interesting things are `treatment` and `sleeptime`, which are the two variables we'll need for our analysis of variance.

■

(c) Using your new data frame, make side-by-side boxplots of sleep time by treatment group.

[8]Sometimes the column playing the role of `rep` *is* interesting to us, but not here.

Solution

```
ggplot(sleep, aes(x = treatment, y = sleeptime)) + geom_boxplot()
```

■

(d) In your boxplots, how does the median sleep time appear to depend on treatment group?

Solution

It appears to *decrease* as the dose of ethanol increases, and pretty substantially so (in that the differences ought to be significant, but that's coming up).

■

(e) There is an assumption about spread that the analysis of variance needs in order to be reliable. Do your boxplots indicate that this assumption is satisfied for these data, bearing in mind that you have only five observations per group?

Solution

The assumption is that the population SDs of each group are all equal. Now, the boxplots show IQRs, which are kind of a surrogate for SD, and because we only have five observations per group to base the IQRs on, the *sample* IQRs might vary a bit. So we should look at the heights of the boxes on the boxplot, and see whether they are grossly unequal. They appear to be to be of very similar heights, all things considered, so I am happy.

If you want the SDs themselves:

```
sleep %>%
  group_by(treatment) %>%
  summarize(stddev = sd(sleeptime))
```

```
# A tibble: 4 x 2
  treatment stddev
  <chr>      <dbl>
1 e0         10.2
2 e1          9.34
3 e2          9.46
4 e4          9.56
```

Those are *very* similar, given only 5 observations per group. No problems here.

■

(f) Run an analysis of variance to see whether sleep time differs significantly among treatment groups. What do you conclude?

Solution

I use `aov` here, because I might be following up with Tukey in a minute:

```
sleep.1 <- aov(sleeptime ~ treatment, data = sleep)
summary(sleep.1)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
treatment    3   5882    1961   21.09 8.32e-06 ***
Residuals   16   1487      93
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is a very small P-value, so my conclusion is that the mean sleep times are not all the same for the treatment groups. Further than that I am not entitled to say (yet).

The technique here is to save the output from `aov` in something, look at that (via `summary`), and then that same something gets fed into `TukeyHSD` later.

■

(g) Would it be a good idea to run Tukey's method here? Explain briefly why or why not, and if you think it would be a good idea, run it.

Solution

Tukey's method is useful when (i) we have run an analysis of variance and got a significant result and (ii) when we want to know which groups differ significantly from which. Both (i) and (ii) are true here. So:

```
TukeyHSD(sleep.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = sleeptime ~ treatment, data = sleep)

$treatment
        diff       lwr         upr     p adj
e1-e0 -17.74 -35.18636  -0.2936428 0.0455781
e2-e0 -31.36 -48.80636 -13.9136428 0.0005142
e4-e0 -46.52 -63.96636 -29.0736428 0.0000056
e2-e1 -13.62 -31.06636   3.8263572 0.1563545
e4-e1 -28.78 -46.22636 -11.3336428 0.0011925
e4-e2 -15.16 -32.60636   2.2863572 0.1005398
```

■

(h) What do you conclude from Tukey's method? (This is liable to be a bit complicated.) Is there a treatment that is clearly best, in terms of the sleep time being largest?

Solution

All the differences are significant except treatment `e2` vs. `e1` and `e4`. All the differences involving the control group `e0` are significant, and if you look back at the boxplots in (c), you'll see that the control group `e0` had the *highest* mean sleep time. So the control group is best (from this point of view), or another way of saying it is that *any* dose of ethanol is significantly reducing mean sleep time. The other comparisons are a bit confusing, because the 1-4 difference is significant, but neither of the differences involving 2 are. That is, 1 is better than 4, but 2 is not significantly worse than 1 nor better than 4. This seems like it should be a logical impossibility, but the story is that we don't have enough data to decide where 2 fits relative to 1 or 4. If we had 10 or 20 observations per group, we might be able to conclude that 2 is in between 1 and 4 as the boxplots suggest.

Extra: I didn't ask about normality here, but like the equal-spreads assumption I'd say there's nothing controversial about it with these data. With normality good and equal spreads good, `aov` plus Tukey is the analysis of choice.

■

15.16 Growth of tomatoes

A biology graduate student exposed each of 32 tomato plants to one of four different colours of light (8 plants to each colour). The growth rate of each plant, in millimetres per week, was recorded. The data are in link.

(a) Read the data into R and confirm that you have 8 rows and 5 columns of data.

Solution

This kind of thing:

```
my_url="http://ritsokiguess.site/datafiles/tomatoes.txt"
toms1=read_delim(my_url," ")
```

```
Rows: 8 Columns: 5
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (5): plant, blue, red, yellow, green

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
toms1
```

```
# A tibble: 8 x 5
  plant  blue   red yellow green
  <dbl> <dbl> <dbl>  <dbl> <dbl>
1     1  5.34  13.7   4.61  2.72
2     2  7.45  13.0   6.63  1.08
3     3  7.15  10.2   5.29  3.97
4     4  5.53  13.1   5.29  2.66
5     5  6.34  11.1   4.76  3.69
6     6  7.16  11.4   5.57  1.96
7     7  7.77  14.0   6.57  3.38
8     8  5.09  13.5   5.25  1.87
```

I do indeed have 8 rows and 5 columns.

With only 8 rows, listing the data like this is good.

■

(b) Re-arrange the data so that you have *one* column containing all the growth rates, and another column saying which colour light each plant was exposed to. (The aim here is to produce something suitable for feeding into `aov` later.)

Solution

This is a job for `pivot_longer`:

```
toms1 %>%
   pivot_longer(-plant, names_to="colour", values_to="growthrate") -> toms2
toms2
```

```
# A tibble: 32 x 3
   plant colour growthrate
   <dbl> <chr>       <dbl>
 1     1 blue         5.34
 2     1 red         13.7
 3     1 yellow       4.61
 4     1 green        2.72
 5     2 blue         7.45
 6     2 red         13.0
 7     2 yellow       6.63
 8     2 green        1.08
 9     3 blue         7.15
10     3 red         10.2
# i 22 more rows
```

I chose to specify "everything but plant number", since there are several colour columns with different names.

Since the column `plant` was never mentioned, this gets repeated as necessary, so now it denotes "plant within colour group", which in this case is not very useful. (Where you have matched pairs, or repeated measures in general, you *do* want to keep track of which individual is which. But this is not repeated measures because plant number 1 in the blue group and plant number 1 in the red group are *different* plants.)

There were 8 rows originally and 4 different colours, so there should be, and are, $8 \times 4 = 32$ rows in the made-longer data set.

■

(c) Save the data in the new format to a text file. This is most easily done using `write_csv`, which is the opposite of `read_csv`. It requires two things: a data frame, and the name of a file to save in, which should have a `.csv` extension.

Solution

The code is easy enough:

```
write_csv(toms2,"tomatoes2.csv")
```

If no error, it worked. That's all you need.

To verify (for my satisfaction) that it was saved correctly:

```
cat tomatoes2.csv
```

```
plant,colour,growthrate
1,blue,5.34
1,red,13.67
1,yellow,4.61
1,green,2.72
2,blue,7.45
2,red,13.04
2,yellow,6.63
2,green,1.08
3,blue,7.15
3,red,10.16
3,yellow,5.29
3,green,3.97
4,blue,5.53
4,red,13.12
4,yellow,5.29
4,green,2.66
5,blue,6.34
5,red,11.06
5,yellow,4.76
5,green,3.69
6,blue,7.16
6,red,11.43
6,yellow,5.57
6,green,1.96
7,blue,7.77
7,red,13.98
7,yellow,6.57
7,green,3.38
8,blue,5.09
8,red,13.49
```

```
8,yellow,5.25
8,green,1.87
```

On my system, that will list the contents of the file. Or you can just open it in R Studio (if you saved it the way I did, it'll be in the same folder, and you can find it in the Files pane.)

■

(d) Make a suitable boxplot, and use it to assess the assumptions for ANOVA. What do you conclude? Explain briefly.

Solution

Nothing terribly surprising here. My data frame is called `toms2`, for some reason:

```
ggplot(toms2,aes(x=colour, y=growthrate))+geom_boxplot()
```

There are no outliers, but there is a little skewness (compare the *whiskers*, not the placement of the median within the box, because what matters with skewness is the *tails*, not the middle of the distribution; it's problems in the tails that make the mean unsuitable as a measure of centre). The Red group looks the most skewed. Also, the Yellow group has smaller spread than the others (we assume that the population variances within each group are equal). The thing to bear in mind here, though, is that there are only eight observations per group, so the distributions could appear to have unequal variances or some non-normality by chance.

My take is that these data, all things considered, are *just about* OK for ANOVA. Another option would be to do Welch's ANOVA as well and compare with the regular ANOVA: if they give more or less the same P-value, that's a sign that I didn't need to worry.

Extra: some people like to run a formal test on the variances to test them for equality. My favourite (for reasons explained elsewhere) is the Levene test, if you insist on going this way. It lives in package `car`, and *does not* take a `data=`, so you need to do the `with` thing:

```
library(car)
with(toms2,leveneTest(growthrate,colour))
```

```
Warning in leveneTest.default(growthrate, colour): colour coerced to factor.
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  3  0.9075 0.4499
      28
```

The warning is because `colour` was actually text, but the test did the right thing by turning it into a factor, so that's OK.

There is no way we can reject equal variances in the four groups. The F-statistic is less than 1, in fact, which says that if the four groups have the same population variances, the sample variances will be *more* different than the ones we observed on average, and so there is no way that these sample variances indicate different population variances. (This is because of 8 observations only per group; if there had been 80 observations per group, it would have been a different story.) Decide for yourself whether you're surprised by this.

With that in mind, I think the regular ANOVA will be perfectly good, and we would expect that and the Welch ANOVA to give very similar results.

■

(e) Run (regular) ANOVA on these data. What do you conclude? (Optional extra: if you think that some other variant of ANOVA would be better, run that as well and compare the results.)

Solution

`aov`, bearing in mind that Tukey is likely to follow:

```
toms.1=aov(growthrate~colour,data=toms2)
summary(toms.1)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
colour       3  410.5  136.82   118.2 5.28e-16 ***
Residuals   28   32.4    1.16
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is a tiny P-value, so the mean growth rate for the different colours is definitely *not* the same for all colours. Or, if you like, one or more of the colours has a different mean growth rate than the others.

This, remember, is as far as we go right now.

Extra: if you thought that normality was OK but not equal spreads, then Welch ANOVA is the way to go:

```r
toms.2=oneway.test(growthrate~colour,data=toms2)
toms.2
```

```
    One-way analysis of means (not assuming equal variances)

data:  growthrate and colour
F = 81.079, num df = 3.000, denom df = 15.227, p-value = 1.377e-09
```

The P-value is not *quite* as small as for the regular ANOVA, but it is still very small, and the conclusion is the same.

If you had doubts about the normality (that were sufficiently great, even given the small sample sizes), then go with Mood's median test for multiple groups:

```r
library(smmr)
median_test(toms2,growthrate,colour)
```

```
$table
        above
group    above below
  blue       5     3
  green      0     8
  red        8     0
  yellow     3     5

$test
```

```
       what        value
1 statistic 1.700000e+01
2        df 3.000000e+00
3   P-value 7.067424e-04
```

The P-value is again extremely small (though not quite as small as for the other two tests, for the usual reason that Mood's median test doesn't use the data very efficiently: it doesn't use how *far* above or below the overall median the data values are.)

The story here, as ever, is consistency: whatever you thought was wrong, looking at the boxplots, needs to guide the test you do:

- if you are not happy with normality, go with `median_test` from `smmr` (Mood's median test).
- if you are happy with normality and equal variances, go with `aov`.
- if you are happy with normality but not equal variances, go with `oneway.test` (Welch ANOVA).

So the first thing to think about is normality, and if you are OK with normality, then think about equal spreads. Bear in mind that you need to be willing to tolerate a certain amount of non-normality and inequality in the spreads, given that your data are only samples from their populations. (Don't expect perfection, in short.)

■

(f) If warranted, run a suitable follow-up. (If not warranted, explain briefly why not.)

Solution

Whichever flavour of ANOVA you ran (regular ANOVA, Welch ANOVA, Mood's median test), you got the same conclusion for these data: that the average growth rates were not all the same for the four colours. That, as you'll remember, is as far as you go. To find out which colours differ from which in terms of growth rate, you need to run some kind of multiple-comparisons follow-up, the right one for the analysis you did. Looking at the boxplots suggests that red is clearly best and green clearly worst, and it is possible that all the colours are significantly different from each other.) If you did regular ANOVA, Tukey is what you need:

```r
TukeyHSD(toms.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = growthrate ~ colour, data = toms2)
```

```
$colour
                diff       lwr        upr     p adj
green-blue   -3.8125 -5.281129 -2.3438706 0.0000006
red-blue      6.0150  4.546371  7.4836294 0.0000000
yellow-blue  -0.9825 -2.451129  0.4861294 0.2825002
red-green     9.8275  8.358871 11.2961294 0.0000000
yellow-green  2.8300  1.361371  4.2986294 0.0000766
yellow-red   -6.9975 -8.466129 -5.5288706 0.0000000
```

All of the differences are (strongly) significant, except for yellow and blue, the two with middling growth rates on the boxplot. Thus we would have no hesitation in saying that growth rate is biggest in red light and smallest in green light.

If you did Welch ANOVA, you need Games-Howell, which you have to get from one of the packages that offers it:

```r
library(PMCMRplus)
gamesHowellTest(growthrate~factor(colour),data=toms2)
```

```
    Pairwise comparisons using Games-Howell test

data: growthrate by factor(colour)

       blue    green   red
green  1.6e-05 -       -
red    1.5e-06 4.8e-09 -
yellow 0.18707 0.00011 5.8e-07


P value adjustment method: none

alternative hypothesis: two.sided
```

The conclusions are the same as for the Tukey: all the means are significantly different except for yellow and blue. Finally, if you did Mood's median test, you need this one:

```r
pairwise_median_test(toms2, growthrate, colour)
```

```
# A tibble: 6 x 4
  g1    g2       p_value adj_p_value
  <chr> <chr>      <dbl>       <dbl>
1 blue  green  0.0000633    0.000380
2 blue  red    0.0000633    0.000380
3 blue  yellow 0.317        1
4 green red    0.0000633    0.000380
5 green yellow 0.0000633    0.000380
6 red   yellow 0.0000633    0.000380
```

Same conclusions again. This is what I would have guessed; the conclusions from Tukey were so clear-cut that it really didn't matter which way you went; you'd come to the same conclusion.

That said, what I am looking for from you is a sensible choice of analysis of variance (ANOVA, Welch's ANOVA or Mood's median test) for a good reason, followed by the *right* follow-up for the test you did. Even though the conclusions are all the same no matter what you do here, I want you to get used to following the right method, so that you will be able to do the right thing when it *does* matter.

■

15.17 Pain relief in migraine headaches (again)

The data in link are from a study of pain relief in migraine headaches. Specifically, 27 subjects were randomly assigned to receive *one* of three pain relieving drugs, labelled A, B and C. Each subject reported the number of hours of pain relief they obtained (that is, the number of hours between taking the drug and the migraine symptoms returning). A higher value is therefore better. Can we make some recommendation about which drug is best for the population of migraine sufferers?

(a) Read in and display the data. Take a look at the data file first, and see if you can say why `read_table` will work and `read_delim` will not.

Solution

The key is two things: the data values are *lined up in columns*, and *there is more than one space between values*. The second thing is why `read_delim` will not work. If you look carefully at the data file, you'll see that the column names are above and aligned with the columns. `read_table` doesn't actually need things to be lined up in columns; all it actually needs is for there to be one or more spaces between columns.

```r
my_url <- "http://ritsokiguess.site/datafiles/migraine.txt"
migraine <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  DrugA = col_double(),
  DrugB = col_double(),
  DrugC = col_double()
)
```

```r
migraine
```

```
# A tibble: 9 x 3
  DrugA DrugB DrugC
  <dbl> <dbl> <dbl>
1     4     6     6
2     5     8     7
3     4     4     6
4     3     5     6
5     2     4     7
6     4     6     5
7     3     5     6
8     4    11     5
9     4    10     5
```

Success.

■

(b) What is it about the experimental design that makes a one-way analysis of variance plausible for data like this?

Solution

Each experimental subject only tested *one* drug, so that we have 27 independent observations, nine from each drug. This is exactly the setup that a one-way ANOVA requires. Compare that to, for example, a situation where you had only 9 subjects, but they each tested *all* the drugs (so that each subject produced three measurements). That is like a three-measurement version of matched pairs, a so-called **repeated-measures design**, which requires its own kind of analysis.[9]

[9] To allow for the fact that measurements on the same subject are not independent but correlated.

■

(c) What is wrong with the current format of the data as far as doing a one-way ANOVA analysis is concerned? (This is related to the idea of whether or not the data are "tidy".)

Solution

For our analysis, we need one column of pain relief time and one column labelling the drug that the subject in question took. Or, if you prefer to think about what would make these data "tidy": there are 27 subjects, so there ought to be 27 rows, and all three columns are measurements of pain relief, so they ought to be in one column.

■

(d) "Tidy" the data to produce a data frame suitable for your analysis.

Solution

This is `pivot_longer`. The column names are going to be stored in a column `drug`, and the corresponding values in a column called `painrelief` (use whatever names you like):

```
migraine %>%
  pivot_longer(everything(), names_to="drug", values_to="painrelief") -> migraine2
```

Since I was making all the columns longer, I used the select-helper `everything()` to do that. Using instead `DrugA:DrugC` or `starts_with("Drug")` would also be good. Try them. `starts_with` is not case-sensitive, as far as I remember, so `starts_with("drug")` will also work here.

We do indeed have a new data frame with 27 rows, one per observation, and 2 columns, one for each variable: the pain relief hours, plus a column identifying which drug that pain relief time came from. Exactly what `aov` needs.

You can probably devise a better name for your new data frame.

■

(e) Go ahead and run your one-way ANOVA (and Tukey if necessary). Assume for this that the pain relief hours in each group are sufficiently close to normally distributed with sufficiently equal spreads.

Solution

My last sentence absolves us from doing the boxplots that we would normally insist on doing.

```
painrelief.1 <- aov(painrelief ~ drug, data = migraine2)
summary(painrelief.1)
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)
drug         2  41.19   20.59   7.831 0.00241 **
Residuals   24  63.11    2.63
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There are (strongly) significant differences among the drugs, so it is definitely worth firing up Tukey to figure out where the differences are:

```r
TukeyHSD(painrelief.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = painrelief ~ drug, data = migraine2)

$drug
                  diff        lwr      upr     p adj
DrugB-DrugA  2.8888889  0.9798731 4.797905 0.0025509
DrugC-DrugA  2.2222222  0.3132065 4.131238 0.0203671
DrugC-DrugB -0.6666667 -2.5756824 1.242349 0.6626647
```

Both the differences involving drug A are significant, and because a high value of `painrelief` is better, in both cases drug A is *worse* than the other drugs. Drugs B and C are not significantly different from each other.

Extra: we can also use the "pipe" to do this all in one go:

```r
migraine %>%
  pivot_longer(everything(), names_to="drug", values_to="painrelief") %>%
  aov(painrelief ~ drug, data = .) %>%
  summary()
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)
drug         2  41.19   20.59   7.831 0.00241 **
Residuals   24  63.11    2.63
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

with the same results as before. Notice that I never actually created a second data frame by name; it was created by `pivot_longer` and then immediately used as input to `aov`.[10] I also used the `data=.` trick to use "the data frame that came out of the previous step" as my input to `aov`.

Read the above like this: "take `migraine`, and then make everything longer, creating new columns `drug` and `painrelief`, and then do an ANOVA of `painrelief` by `drug`, and then summarize the results."

What is even more alarming is that I can feed the output from `aov` straight into `TukeyHSD`:

```
migraine %>%
  pivot_longer(everything(), names_to="drug", values_to="painrelief") %>%
  aov(painrelief ~ drug, data = .) %>%
  TukeyHSD()
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = painrelief ~ drug, data = .)

$drug
                   diff        lwr      upr     p adj
DrugB-DrugA  2.8888889  0.9798731 4.797905 0.0025509
DrugC-DrugA  2.2222222  0.3132065 4.131238 0.0203671
DrugC-DrugB -0.6666667 -2.5756824 1.242349 0.6626647
```

I wasn't sure whether this would work, since the output from `aov` is an R `list` rather than a data frame, but the output from `aov` is sent into `TukeyHSD` whatever kind of thing it is.

What I am missing here is to display the result of `aov` *and* use it as input to `TukeyHSD`. Of course, I had to discover that this could be solved, and indeed it can:

```
migraine %>%
  pivot_longer(everything(), names_to="drug", values_to="painrelief") %>%
  aov(painrelief ~ drug, data = .) %>%
  {
    print(summary(.))
    .
  } %>%
  TukeyHSD()
```

[10] And then thrown away.

```
            Df Sum Sq Mean Sq F value  Pr(>F)
drug         2  41.19   20.59   7.831 0.00241 **
Residuals   24  63.11    2.63
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = painrelief ~ drug, data = .)

$drug
                  diff        lwr      upr     p adj
DrugB-DrugA  2.8888889  0.9798731 4.797905 0.0025509
DrugC-DrugA  2.2222222  0.3132065 4.131238 0.0203671
DrugC-DrugB -0.6666667 -2.5756824 1.242349 0.6626647
```

The odd-looking second-last line of that again uses the `.` trick for "whatever came out of the previous step". The thing inside the curly brackets is two commands one after the other; the first is to display the `summary` of that `aov`[11] and the second is to just pass whatever came out of the previous line, the output from `aov`, on, unchanged, into `TukeyHSD`.

In the Unix/Linux world this is called `tee`, where you print something *and* pass it on to the next step. The name `tee` comes from a (real physical) pipe that plumbers would use to split water flow into two, which looks like a letter T.

■

(f) What recommendation would you make about the best drug or drugs? Explain briefly.

Solution

Drug A is significantly the worst, so we eliminate that. But there is no significant difference between drugs B and C, so we have no reproducible reason for preferring one rather than the other. Thus, we recommend "either B or C". If you weren't sure which way around the drugs actually came out, then you should work out the mean pain relief score by drug:

```r
migraine2 %>%
  group_by(drug) %>%
  summarize(m = mean(painrelief))
```

[11] It needs `print` around it to display it, as you need `print` to display something within a loop or a function.

```
# A tibble: 3 x 2
  drug      m
  <chr> <dbl>
1 DrugA  3.67
2 DrugB  6.56
3 DrugC  5.89
```

These confirm that A is worst, and there is nothing much to choose between B and C. You should *not* recommend drug B over drug C on this evidence, just because its (sample) mean is higher. The point about significant differences is that they are supposed to stand up to replication: in another experiment, or in real-life experiences with these drugs, the mean pain relief score for drug A is expected to be worst, but between drugs B and C, sometimes the mean of B will come out higher and sometimes C's mean will be higher, because there is no significant difference between them.[12] Another way is to draw a boxplot of pain-relief scores:

```
ggplot(migraine2, aes(x = drug, y = painrelief)) + geom_boxplot()
```

The medians of drugs B and C are actually exactly the same. Because the pain relief values are all whole numbers (and there are only 9 in each group), you get that thing where enough

[12]This talks about *means* rather than individual observations; in individual cases, sometimes even drug *A* will come out best. But we're interested in population means, since we want to do the greatest good for the greatest number. "Greatest good for the greatest number" is from Jeremy Bentham, 1748–1832, British philosopher and advocate of utilitarianism.

of them are equal that the median and third quartiles are equal, actually for two of the three groups.

Despite the weird distributions, I'm willing to call these groups sufficiently symmetric for the ANOVA to be OK, but I didn't ask you to draw the boxplot, because I didn't want to confuse the issue with this. The point of this question was to get the data tidy enough to do an analysis.

As I said, I didn't want you to have to get into this, but if you are worried, you know what the remedy is — Mood's median test. Don't forget to use the right data frame:

```
library(smmr)
median_test(migraine2, painrelief, drug)
```

```
$table
       above
group   above below
  DrugA     0     8
  DrugB     5     2
  DrugC     6     0

$test
       what        value
1 statistic 1.527273e+01
2        df 2.000000e+00
3   P-value 4.825801e-04
```

Because the pain relief scores are integers, there are probably a lot of them equal to the overall median. There were 27 observations altogether, but Mood's median test will discard any that are equal to this value. There must have been 9 observations in each group to start with, but if you look at each row of the table, there are only 8 observations listed for drug A, 7 for drug B and 6 for drug C, so there must have been 1, 2 and 3 (totalling 6) observations equal to the median that were discarded.

The P-value is a little bigger than came out of the F-test, but the conclusion is still that there are definitely differences among the drugs in terms of pain relief. The table at the top of the output again suggests that drug A is worse than the others, but to confirm that you'd have to do Mood's median test on all three *pairs* of drugs, and then use Bonferroni to allow for your having done three tests:

```
pairwise_median_test(migraine2, painrelief, drug)
```

```
# A tibble: 3 x 4
  g1    g2     p_value adj_p_value
  <chr> <chr>    <dbl>       <dbl>
1 DrugA DrugB 0.00721     0.0216
2 DrugA DrugC 0.000183    0.000548
3 DrugB DrugC 0.921       1
```

Drug A gives worse pain relief (fewer hours) than both drugs B and C, which are not significantly different from each hour. This is exactly what you would have guessed from the boxplot.

I adjusted the P-values as per Bonferroni by multiplying them by 3 (so that I could still compare with 0.05), but it makes no sense to have a P-value, which is a probability, greater than 1, so an "adjusted P-value" that comes out greater than 1 is rounded back down to 1. You interpret this as being "no evidence at all of a difference in medians" between drugs B and C.

■

15.18 Location, species and disease in plants

The table below is a "contingency table", showing frequencies of diseased and undiseased plants of two different species in two different locations:

```
Species     Disease present         Disease absent
          Location X Location Y  Location X Location Y
A            44         12          38        10
B            28         22          20        18
```

The data were saved as link. In that file, the columns are coded by two letters: a `p` or an `a` to denote presence or absence of disease, and an `x` or a `y` to denote location X or Y. The data are separated by multiple spaces and aligned with the variable names.

(a) Read in and display the data.

Solution

`read_table` again. You know this because, when you looked at the data file, which of course you did (didn't you?), you saw that the data values were aligned by columns with multiple spaces between them:

```r
my_url <- "http://ritsokiguess.site/datafiles/disease.txt"
tbl <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  Species = col_character(),
  px = col_double(),
  py = col_double(),
  ax = col_double(),
  ay = col_double()
)
```

```r
tbl
```

```
# A tibble: 2 x 5
  Species    px    py    ax    ay
  <chr>   <dbl> <dbl> <dbl> <dbl>
1 A          44    12    38    10
2 B          28    22    20    18
```

I was thinking ahead, since I'll be wanting to have one of my columns called `disease`, so I'm *not* calling the data frame `disease`.

You'll also have noticed that I simplified the data frame that I had you read in, because the original contingency table I showed you has *two* header rows, and we have to have *one* header row. So I mixed up the information in the two header rows into one.

■

(b) Explain briefly how these data are not "tidy".

Solution

The simple answer is that there are 8 frequencies, that each ought to be in a row by themselves. Or, if you like, there are three variables, Species, Disease status and Location, and each of *those* should be in a *column* of its own. Either one of these ideas, or something like it, is good. I need you to demonstrate that you know something about "tidy data" in this context.

■

(c) Use a suitable `tidyr` tool to get all the things that are the same into a single column. (You'll need to make up a temporary name for the other new column that you create.) Show your result.

Solution

`pivot_longer` is the tool. All the columns apart from `Species` contain frequencies. They are frequencies in disease-location combinations, so I'll call the column of "names" `disloc`. Feel free to call it `temp` for now if you prefer:

```
tbl %>% pivot_longer(-Species, names_to="disloc", values_to = "frequency") -> tbl.2
tbl.2
```

```
# A tibble: 8 x 3
  Species disloc frequency
  <chr>   <chr>      <dbl>
1 A       px            44
2 A       py            12
3 A       ax            38
4 A       ay            10
5 B       px            28
6 B       py            22
7 B       ax            20
8 B       ay            18
```

■

(d) Explain briefly how the data frame you just created is still not "tidy" yet.

Solution

The column I called `disloc` actually contains *two* variables, disease and location, which need to be split up. A check on this is that we have two columns (not including the frequencies), but back in (b) we found *three* variables, so there ought to be three non-frequency columns.

■

(e) Use one more `tidyr` tool to make these data tidy, and show your result.

Solution

This means splitting up `disloc` into two separate columns, splitting after the first character, thus:

```
(tbl.2 %>% separate(disloc, c("disease", "location"), 1) -> tbl.3)
```

```
# A tibble: 8 x 4
  Species disease location frequency
  <chr>   <chr>   <chr>        <dbl>
1 A       p       x               44
2 A       p       y               12
3 A       a       x               38
4 A       a       y               10
5 B       p       x               28
6 B       p       y               22
7 B       a       x               20
8 B       a       y               18
```

This is now tidy: eight frequencies in rows, and three non-frequency columns. (Go back and look at your answer to part (b) and note that the issues you found there have all been resolved now.)

Extra: my reading of one of the vignettes (the one called `pivot`) for `tidyr` suggests that `pivot_longer` can do both the making longer and the separating in one shot:

```
tbl %>% pivot_longer(-Species, names_to=c("disease", "location"), names_sep=1, values_to="
```

```
# A tibble: 8 x 4
  Species disease location frequency
  <chr>   <chr>   <chr>        <dbl>
1 A       p       x               44
2 A       p       y               12
3 A       a       x               38
4 A       a       y               10
5 B       p       x               28
6 B       p       y               22
7 B       a       x               20
8 B       a       y               18
```

And I (amazingly) got that right first time!

The idea is that you recognize that the column names are actually two things: a disease status and a location. To get `pivot_longer` to recognize that, you put two things in the `names_to`. Then you have to say how the two things in the columns are separated: this might be by an underscore or a dot, or, as here, "after the first character" (just as in `separate`). Using two names and some indication of what separates them then does a combined pivot-longer-and-separate, all in one shot.

The more I use `pivot_longer`, the more I marvel at the excellence of its design: it seems to be easy to guess how to make things work.

■

(f) Let's see if we can re-construct the original contingency table (or something equivalent to it). Use the function `xtabs`. This requires first a model formula with the frequency variable on the left of the squiggle, and the other variables separated by plus signs on the right. Second it requires a data frame, with `data=`. Feed your data frame from the previous part into `xtabs`. Save the result in a variable and display the result.

Solution

```
tbl.4 <- xtabs(frequency ~ Species + disease + location, data = tbl.3)
tbl.4
```

```
, , location = x

       disease
Species  a  p
      A 38 44
      B 20 28

, , location = y

       disease
Species  a  p
      A 10 12
      B 18 22
```

This shows a pair of contingency tables, one each for each of the two locations (in general, the variable you put last on the right side of the model formula). You can check that everything corresponds with the original data layout at the beginning of the question, possibly with some things rearranged (but with the same frequencies in the same places).

■

(g) Take the output from the last part and feed it into the function `ftable`. How has the output been changed? Which do you like better? Explain briefly.

Solution

This:

```r
ftable(tbl.4)
```

```
                location  x  y
Species disease
A       a               38 10
        p               44 12
B       a               20 18
        p               28 22
```

This is the same output, but shown more compactly. (Rather like a vertical version of the original data, in fact.) I like `ftable` better because it displays the data in the smallest amount of space, though I'm fine if you prefer the `xtabs` output because it spreads things out more. This is a matter of taste. Pick one and tell me why you prefer it, and I'm good.

That's the end of what you had to do, but I thought I would do some modelling and try to find out what's associated with disease. The appropriate modelling with frequencies is called "log-linear modelling", and it assumes that the log of the frequencies has a linear relationship with the effects of the other variables. This is not quite as simple as the log transformations we had before, because bigger frequencies are going to be more variable, so we fit a generalized linear model with a Poisson-distributed response and log link. (It's better if you know what that means, but you ought to be able to follow the logic if you don't. Chapter 29 has more on this.)

First, fit a model predicting frequency from everything, including all the interactions. (The reason for doing it this way will become clear later):

```r
model.1 <- glm(frequency ~ Species * location * disease, data = tbl.3, family = "poisson")
drop1(model.1, test = "Chisq")
```

```
Single term deletions

Model:
frequency ~ Species * location * disease
                        Df Deviance    AIC      LRT Pr(>Chi)
<none>                     0.000000 55.291
Species:location:disease  1 0.070257 53.362 0.070257    0.791
```

The residuals are all zero because this model fits perfectly. The problem is that it is very complicated, so it offers no insight. So what we do is to look at the highest-order interaction `Species:location:disease` and see whether it is significant. It is not, so we can remove it. This is reminiscent of variable selection in regression, where we pull the least significant thing

out of the model in turn until we can go no further. But here, we have additional things to think about: we have to get rid of all the three-way interactions before we can tackle the two-way ones, and all the two-way ones before we can tackle the main effects. There is a so-called "nested" structure happening here that says you don't look at, say, `Species`, until you have removed *all* the higher-order interactions involving `Species`. Not clear yet? Don't fret. `drop1` allows you to assess what is currently up for grabs (here, only the three-way interaction, which is not significant, so out it comes).

Let's get rid of that three-way interaction. This is another use for `update` that you might have seen in connection with multiple regression (to make small changes to a big model):

```
model.2 <- update(model.1, . ~ . - Species:location:disease)
drop1(model.2, test = "Chisq")
```

```
Single term deletions

Model:
frequency ~ Species + location + disease + Species:location +
    Species:disease + location:disease
                 Df Deviance    AIC     LRT  Pr(>Chi)
<none>                0.0703 53.362
Species:location  1  13.0627 64.354 12.9924 0.0003128 ***
Species:disease   1   0.2696 51.561  0.1993 0.6552865
location:disease  1   0.1043 51.396  0.0340 0.8536877
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice how `update` saved us having to write the whole model out again.

Now the three two-way interactions are up for grabs: `Species:location`, `Species:disease` and `location:disease`. The last of these is the least significant, so out it comes. I did some copying and pasting, but I had to remember which model I was working with and what I was removing:

```
model.3 <- update(model.2, . ~ . - location:disease)
drop1(model.3, test = "Chisq")
```

```
Single term deletions

Model:
frequency ~ Species + location + disease + Species:location +
    Species:disease
```

```
                 Df Deviance    AIC     LRT  Pr(>Chi)
<none>                0.1043 51.396
Species:location  1  13.0678 62.359 12.9635 0.0003176 ***
Species:disease   1   0.2746 49.566  0.1703 0.6798021
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`Species:disease` comes out, but it looks as if `Species:location` will have to stay:

```r
model.4 <- update(model.3, . ~ . - Species:disease)
drop1(model.4, test = "Chisq")
```

```
Single term deletions

Model:
frequency ~ Species + location + disease + Species:location
                 Df Deviance    AIC     LRT  Pr(>Chi)
<none>                0.2746 49.566
disease           1   2.3617 49.653  2.0871 0.1485461
Species:location  1  13.2381 60.530 12.9635 0.0003176 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`Species:location` indeed stays. That means that anything "contained in" it also has to stay, regardless of its main effect. So the only candidate for removal now is `disease`: not significant, out it comes:

```r
model.5 <- update(model.4, . ~ . - disease)
drop1(model.5, test = "Chisq")
```

```
Single term deletions

Model:
frequency ~ Species + location + Species:location
                 Df Deviance    AIC    LRT  Pr(>Chi)
<none>                2.3617 49.653
Species:location  1  15.3252 60.617 12.963 0.0003176 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And now we have to stop.

What does this final model mean? Well, frequency depends significantly on the `Species:location` combination, but not on anything else. To see how, we make a contingency table of species by location (totalling up over disease status, since that is not significant):

```
xtabs(frequency ~ Species + location, data = tbl.3)
```

```
       location
Species  x  y
      A 82 22
      B 48 40
```

Most of the species A's are at location X, but the species B's are about evenly divided between the two locations. Or, if you prefer (equally good): location X has mostly species A, while location Y has mostly species B. You can condition on either variable and compare the conditional distribution of the other one.

Now, this is rather interesting, because this began as a study of disease, but disease has completely disappeared from our final model! That means that nothing in our final model has any relationship with disease. Indeed, if you check the original table, you'll find that disease is present slightly more than it's absent, for all combinations of species and location. That is, neither species nor location has any particular association with (effect on) disease, since disease prevalence doesn't change appreciably if you change location, species or the combination of them.

The way an association with disease would show up is if a `disease:`something interaction had been significant and had stayed in the model, that something would have been associated with disease. For example, if the `disease:Species` table had looked like this:

```
disease <- c("a", "a", "p", "p")
Species <- c("A", "B", "A", "B")
frequency <- c(10, 50, 30, 30)
xx <- tibble(disease, Species, frequency)
xtabs(frequency ~ disease + Species, data=xx)
```

```
       Species
disease  A  B
      a 10 50
      p 30 30
```

For species A, disease is present 75% of the time, but for species B it's present less than 40% of the time. So in this one there ought to be a significant association between disease and species:

```
xx.1 <- glm(frequency ~ disease * Species, data = xx, family = "poisson")
drop1(xx.1, test = "Chisq")
```

```
Single term deletions

Model:
frequency ~ disease * Species
                Df Deviance    AIC    LRT  Pr(>Chi)
<none>               0.000 28.400
disease:Species  1   15.518 41.918 15.518 8.171e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And so there is. Nothing can come out of the model. (This is the same kind of test as a chi-squared test for association.
The log-linear model is a multi-variable generalization of that.)

■

15.19 Mating songs in crickets

Male tree crickets produce "mating songs" by rubbing their wings together to produce a chirping sound. It is hypothesized that female tree crickets identify males of the correct species by how fast (in chirps per second) the male's mating song is. This is called the "pulse rate". Some data for two species of crickets are in link. The columns, which are unlabelled, are temperature and pulse rate (respectively) for *Oecanthus exclamationis* (first two columns) and *Oecanthus niveus* (third and fourth columns). The columns are separated by tabs. There are some missing values in the first two columns because fewer *exclamationis* crickets than *niveus* crickets were measured. The research question is whether males of the different species have different average pulse rates. It is also of interest to see whether temperature has an effect, and if so, what. Before we get to that, however, we have some data organization to do.

(a) Read in the data, allowing for the fact that you have no column names. You'll see that the columns have names `X1` through `X4`. This is OK.

Solution

Tab-separated, so `read_tsv`; no column names, so `col_names=FALSE`:

```r
my_url <- "http://ritsokiguess.site/datafiles/crickets.txt"
crickets <- read_tsv(my_url, col_names = FALSE)
```

```
Rows: 17 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: "\t"
dbl (4): X1, X2, X3, X4

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
crickets
```

```
# A tibble: 17 x 4
      X1    X2    X3    X4
   <dbl> <dbl> <dbl> <dbl>
 1  20.8  67.9  17.2  44.3
 2  20.8  65.1  18.3  47.2
 3  24    77.3  18.3  47.6
 4  24    78.7  18.3  49.6
 5  24    79.4  18.9  50.3
 6  24    80.4  18.9  51.8
 7  26.2  85.8  20.4  60
 8  26.2  86.6  21    58.5
 9  26.2  87.5  21    58.9
10  26.2  89.1  22.1  60.7
11  28.4  98.6  23.5  69.8
12  29   101.   24.2  70.9
13  30.4  99.3  25.9  76.2
14  30.4 102.   26.5  76.1
15  NA    NA    26.5  77
16  NA    NA    26.5  77.7
17  NA    NA    28.6  84.7
```

As promised.

If you didn't catch the tab-separated part, this probably happened to you:

```r
d <- read_delim(my_url, " ", col_names = FALSE)
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for details,
e.g.:
  dat <- vroom(...)
  problems(dat)

Rows: 17 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): X1

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

This doesn't look good:

```r
problems(d)
```

```
# A tibble: 3 x 5
    row   col expected  actual    file
  <int> <int> <chr>     <chr>     <chr>
1    15     2 1 columns 2 columns ""
2    16     2 1 columns 2 columns ""
3    17     2 1 columns 2 columns ""
```

The "expected columns" being 1 should bother you, since we know there are supposed to be 4 columns. At this point, we take a look at what got read in:

```r
d
```

```
# A tibble: 17 x 1
   X1
   <chr>
 1 "20.8\t67.9\t17.2\t44.3"
 2 "20.8\t65.1\t18.3\t47.2"
 3 "24.0\t77.3\t18.3\t47.6"
 4 "24.0\t78.7\t18.3\t49.6"
 5 "24.0\t79.4\t18.9\t50.3"
 6 "24.0\t80.4\t18.9\t51.8"
 7 "26.2\t85.8\t20.4\t60.0"
 8 "26.2\t86.6\t21.0\t58.5"
```

```
 9 "26.2\t87.5\t21.0\t58.9"
10 "26.2\t89.1\t22.1\t60.7"
11 "28.4\t98.6\t23.5\t69.8"
12 "29.0\t100.8\t24.2\t70.9"
13 "30.4\t99.3\t25.9\t76.2"
14 "30.4\t101.7\t26.5\t76.1"
15 "NA\tNA \t26.5\t77.0"
16 "NA\tNA \t26.5\t77.7"
17 "NA\tNA \t28.6\t84.7"
```

and there you see the `\t` or "tab" characters separating the values, instead of spaces. (This is what I tried first, and once I looked at this, I realized that `read_tsv` was what I needed.)

■

(b) Tidy these untidy data, going as directly as you can to something tidy. (Some later parts show you how it used to be done.) Begin by: (i) adding a column of row numbers, (ii) `rename`-ing the columns to species name, an underscore, and the variable contents (keeping `pulserate` as one word), and then use `pivot_longer`. Note that the column names encode *two* things.

Solution

Take this one piece of the pipeline at a time: that is, first check that you got the renaming right and looking at what you have, before proceeding to the `pivot_longer`. The syntax of `rename` is new name equals old name, and I like to split this over several lines to make it easier to read:

```r
crickets %>%
  mutate(row=row_number()) %>%
  rename(
    exclamationis_temperature = X1,
    exclamationis_pulserate = X2,
    niveus_temperature = X3,
    niveus_pulserate = X4
  )
```

```
# A tibble: 17 x 5
   exclamationis_temperature exclamationis_pulserate niveus_temperature
                       <dbl>                   <dbl>              <dbl>
 1                      20.8                    67.9               17.2
 2                      20.8                    65.1               18.3
 3                      24                      77.3               18.3
```

```
 4                      24                      78.7                18.3
 5                      24                      79.4                18.9
 6                      24                      80.4                18.9
 7                      26.2                    85.8                20.4
 8                      26.2                    86.6                21
 9                      26.2                    87.5                21
10                      26.2                    89.1                22.1
11                      28.4                    98.6                23.5
12                      29                     101.                 24.2
13                      30.4                    99.3                25.9
14                      30.4                   102.                 26.5
15                      NA                      NA                  26.5
16                      NA                      NA                  26.5
17                      NA                      NA                  28.6
# i 2 more variables: niveus_pulserate <dbl>, row <int>
```

The first part of each column name is the species and the second part is what was measured each time, separated by an underscore. To handle that in `pivot_longer`, you give *two* names of new columns to create (in `names_to`), and say what they're separated by:

```
crickets %>%
  mutate(row=row_number()) %>%
  rename(
    exclamationis_temperature = X1,
    exclamationis_pulserate = X2,
    niveus_temperature = X3,
    niveus_pulserate = X4
  ) %>%
  pivot_longer(-row, names_to=c("species", "measurement"), names_sep="_", values_to = "obs
```

```
# A tibble: 68 x 4
     row species       measurement   obs
   <int> <chr>         <chr>       <dbl>
 1     1 exclamationis temperature  20.8
 2     1 exclamationis pulserate    67.9
 3     1 niveus        temperature  17.2
 4     1 niveus        pulserate    44.3
 5     2 exclamationis temperature  20.8
 6     2 exclamationis pulserate    65.1
 7     2 niveus        temperature  18.3
 8     2 niveus        pulserate    47.2
 9     3 exclamationis temperature  24
```

```
10     3 exclamationis pulserate    77.3
# i 58 more rows
```

This is tidy now, but we went a step too far: that column `measurement` should be *two* columns, called `temperature` and `pulserate`, which means it should be made wider. The obvious way is this:

```
crickets %>%
  mutate(row=row_number()) %>%
  rename(
    exclamationis_temperature = X1,
    exclamationis_pulserate = X2,
    niveus_temperature = X3,
    niveus_pulserate = X4
  ) %>%
  pivot_longer(-row, names_to=c("species", "measurement"), names_sep="_", values_to = "obs
  pivot_wider(names_from=measurement, values_from=obs)
```

```
# A tibble: 34 x 4
     row species       temperature pulserate
   <int> <chr>               <dbl>     <dbl>
 1     1 exclamationis        20.8      67.9
 2     1 niveus               17.2      44.3
 3     2 exclamationis        20.8      65.1
 4     2 niveus               18.3      47.2
 5     3 exclamationis        24        77.3
 6     3 niveus               18.3      47.6
 7     4 exclamationis        24        78.7
 8     4 niveus               18.3      49.6
 9     5 exclamationis        24        79.4
10     5 niveus               18.9      50.3
# i 24 more rows
```

The row numbers are cricket-within-species, which isn't very meaningful, but we needed something for the `pivot_wider` to key on, to recognize what needed to go in which row. The way it works is it uses anything not mentioned in `names_from` or `values_from` as a "key": each unique combination belongs in a row. Here that would be the combination of row and species, which is a good key because each species appears once with each row number.

This works, but a better way is to recognize that one of the variants of `pivot_longer` will do this all at once (something to think about when you have a longer followed by a wider). The key is that temperature and pulse rate need to be column names, so the second thing in

`names_to` has to be that special thing `.value`, and you remove the `values_to` since it is now clear where the values are coming from:

```
crickets %>%
  mutate(row=row_number()) %>%
  rename(
    exclamationis_temperature = X1,
    exclamationis_pulserate = X2,
    niveus_temperature = X3,
    niveus_pulserate = X4
  ) %>%
  pivot_longer(-row, names_to=c("species", ".value"), names_sep="_")
```

```
# A tibble: 34 x 4
     row species       temperature pulserate
   <int> <chr>               <dbl>     <dbl>
 1     1 exclamationis        20.8      67.9
 2     1 niveus               17.2      44.3
 3     2 exclamationis        20.8      65.1
 4     2 niveus               18.3      47.2
 5     3 exclamationis        24        77.3
 6     3 niveus               18.3      47.6
 7     4 exclamationis        24        78.7
 8     4 niveus               18.3      49.6
 9     5 exclamationis        24        79.4
10     5 niveus               18.9      50.3
# i 24 more rows
```

■

(c) If you found (b) a bit much to take in, the rest of the way we take a rather more leisurely approach towards the tidying.

These data are rather far from being tidy. There need to be three variables, temperature, pulse rate and species, and there are $14 + 17 = 31$ observations altogether. This one is tricky in that there are temperature and pulse rate for each of two levels of a factor, so I'll suggest combining the temperature and chirp rate together into one thing for each species, then pivoting them longer ("combining"), then pivoting them wider again ("splitting"). Create new columns, named for each species, that contain the temperature and pulse rate for that species in that order, `united` together. For the rest of this question, start from the data frame you read in, and build a pipe, one or two steps at a time, to save creating a lot of temporary data frames.

Solution

Breathe, and then begin. `unite` creates new columns by joining together old ones:[13]

```
crickets %>%
  unite(exclamationis, X1:X2) %>%
  unite(niveus, X3:X4)
```

```
# A tibble: 17 x 2
   exclamationis niveus
   <chr>         <chr>
 1 20.8_67.9     17.2_44.3
 2 20.8_65.1     18.3_47.2
 3 24_77.3       18.3_47.6
 4 24_78.7       18.3_49.6
 5 24_79.4       18.9_50.3
 6 24_80.4       18.9_51.8
 7 26.2_85.8     20.4_60
 8 26.2_86.6     21_58.5
 9 26.2_87.5     21_58.9
10 26.2_89.1     22.1_60.7
11 28.4_98.6     23.5_69.8
12 29_100.8      24.2_70.9
13 30.4_99.3     25.9_76.2
14 30.4_101.7    26.5_76.1
15 NA_NA         26.5_77
16 NA_NA         26.5_77.7
17 NA_NA         28.6_84.7
```

Note that the original columns `X1:X4` are *gone*, which is fine, because the information we needed from them is contained in the two new columns. `unite` by default uses an underscore to separate the joined-together values, which is generally safe since you won't often find those in data.

Digression: `unite`-ing with a space could cause problems if the data values have spaces in them already. Consider this list of names:

```
names <- c("Cameron McDonald", "Durwin Yang", "Ole Gunnar Solskjaer", "Mahmudullah")
```

Two very former students of mine, a Norwegian soccer player, and a Bangladeshi cricketer. Only one of these has played for Manchester United:

[13] As `str_c` or `paste` do, actually, but the advantage of `unite` is that it gets rid of the other columns, which you probably no longer need.

```r
manu <- c(F, F, T, F)
```

and let's make a data frame:

```r
d <- tibble(name = names, manu = manu)
d
```

```
# A tibble: 4 x 2
  name                 manu
  <chr>                <lgl>
1 Cameron McDonald     FALSE
2 Durwin Yang          FALSE
3 Ole Gunnar Solskjaer TRUE
4 Mahmudullah          FALSE
```

Now, what happens if we `unite` those columns, separating them by a space?

```r
d %>% unite(joined, name:manu, sep = " ")
```

```
# A tibble: 4 x 1
  joined
  <chr>
1 Cameron McDonald FALSE
2 Durwin Yang FALSE
3 Ole Gunnar Solskjaer TRUE
4 Mahmudullah FALSE
```

If we then try to separate them again, what happens?

```r
d %>%
  unite(joined, name:manu, sep = " ") %>%
  separate(joined, c("one", "two"), " ")
```

```
Warning: Expected 2 pieces. Additional pieces discarded in 3 rows [1, 2, 3].
```

```
# A tibble: 4 x 2
  one         two
  <chr>       <chr>
1 Cameron     McDonald
```

```
2 Durwin      Yang
3 Ole         Gunnar
4 Mahmudullah FALSE
```

Things have gotten lost: most of the original values of `manu` and some of the names. If we use a different separator character, either choosing one deliberately or going with the default underscore, everything works swimmingly:

```
d %>%
  unite(joined, name:manu, sep = ":") %>%
  separate(joined, c("one", "two"), ":")
```

```
# A tibble: 4 x 2
  one                  two
  <chr>                <chr>
1 Cameron McDonald     FALSE
2 Durwin Yang          FALSE
3 Ole Gunnar Solskjaer TRUE
4 Mahmudullah          FALSE
```

and we are back to where we started.

If you run just the `unite` line (move the pipe symbol to the next line so that the `unite` line is complete as it stands), you'll see what happened.

■

(d) The two columns `exclamationis` and `niveus` that you just created are both temperature-pulse rate combos, but for different species. Collect them together into one column, labelled by species. (This is a straight `tidyr pivot_longer`, even though the columns contain something odd-looking.)

Solution

Thus, this, naming the new column `temp_pulse` since it contains both of those things. Add to the end of the pipe you started building in the previous part:

```
crickets %>%
  unite(exclamationis, X1:X2) %>%
  unite(niveus, X3:X4) %>%
  pivot_longer(exclamationis:niveus, names_to = "species", values_to = "temp_pulse")
```

```
# A tibble: 34 x 2
   species       temp_pulse
   <chr>         <chr>
 1 exclamationis 20.8_67.9
 2 niveus        17.2_44.3
 3 exclamationis 20.8_65.1
 4 niveus        18.3_47.2
 5 exclamationis 24_77.3
 6 niveus        18.3_47.6
 7 exclamationis 24_78.7
 8 niveus        18.3_49.6
 9 exclamationis 24_79.4
10 niveus        18.9_50.3
# i 24 more rows
```

Yep. You'll see both species of crickets, and you'll see some missing values at the bottom, labelled, at the moment, `NA_NA`.

This is going to get rather long, but don't fret: we debugged the two `unite` lines before, so if you get any errors, they must have come from the `pivot_longer`. So that would be the place to check.

■

(e) Now split up the temperature-pulse combos at the underscore, into two separate columns. This is `separate`. When specifying what to separate by, you can use a number ("split after this many characters") or a piece of text, in quotes ("when you see this text, split at it").

Solution

The text to split by is an underscore (in quotes), since `unite` by default puts an underscore in between the values it pastes together. Glue the `separate` onto the end. We are creating two new variables `temperature` and `pulse_rate`:

```
crickets %>%
  unite(exclamationis, X1:X2) %>%
  unite(niveus, X3:X4) %>%
  pivot_longer(exclamationis:niveus, names_to = "species", values_to = "temp_pulse") %>%
  separate(temp_pulse, c("temperature", "pulse_rate"), "_")
```

```
# A tibble: 34 x 3
   species       temperature pulse_rate
   <chr>         <chr>       <chr>
```

```
 1 exclamationis 20.8        67.9
 2 niveus        17.2        44.3
 3 exclamationis 20.8        65.1
 4 niveus        18.3        47.2
 5 exclamationis 24          77.3
 6 niveus        18.3        47.6
 7 exclamationis 24          78.7
 8 niveus        18.3        49.6
 9 exclamationis 24          79.4
10 niveus        18.9        50.3
# i 24 more rows
```

You'll note that `unite` and `separate` are opposites ("inverses") of each other, but we haven't just done something and then undone it, because we have a `pivot_longer` in between; in fact, arranging it this way has done precisely the tidying we wanted.

■

(f) Almost there. Temperature and pulse rate are still text (because `unite` turned them into text), but they should be numbers. Create new variables that are numerical versions of temperature and pulse rate (using `as.numeric`). Check that you have no extraneous variables (and, if necessary, get rid of the ones you don't want). (Species is also text and really ought to be a factor, but having it as text doesn't seem to cause any problems.) You can, if you like, use `parse_number` instead of `as.numeric`. They should both work. The distinction I prefer to make is that `parse_number` is good for text with a number in it (that we want to pull the number out of), while `as.numeric` is for turning something that looks like a number but isn't one into a genuine number.[14]

Solution

`mutate`-ing into a column that already exists overwrites the variable that's already there (which saves us some effort here).

```
crickets %>%
  unite(exclamationis, X1:X2) %>%
  unite(niveus, X3:X4) %>%
  pivot_longer(exclamationis:niveus, names_to = "species", values_to = "temp_pulse") %>%
  separate(temp_pulse, c("temperature", "pulse_rate"), "_") %>%
  mutate(temperature = as.numeric(temperature)) %>%
  mutate(pulse_rate = as.numeric(pulse_rate)) -> crickets.1
```

[14] You could just as well make the point that the text 20.8 contains the number 20.8 and nothing else, so that parsing it as text in search of a number will pull out 20.8 as a number. If that logic works for you, go with it.

```
Warning: There was 1 warning in `mutate()`.
i In argument: `temperature = as.numeric(temperature)`.
Caused by warning:
! NAs introduced by coercion

Warning: There was 1 warning in `mutate()`.
i In argument: `pulse_rate = as.numeric(pulse_rate)`.
Caused by warning:
! NAs introduced by coercion
```

```
crickets.1
```

```
# A tibble: 34 x 3
   species       temperature pulse_rate
   <chr>               <dbl>      <dbl>
 1 exclamationis        20.8       67.9
 2 niveus               17.2       44.3
 3 exclamationis        20.8       65.1
 4 niveus               18.3       47.2
 5 exclamationis        24         77.3
 6 niveus               18.3       47.6
 7 exclamationis        24         78.7
 8 niveus               18.3       49.6
 9 exclamationis        24         79.4
10 niveus               18.9       50.3
# i 24 more rows
```

I saved the data frame this time, since this is the one we will use for our analysis.

The warning message tells us that we got genuine missing-value NAs back, which is probably what we want. Specifically, they got turned from missing *text* to missing *numbers*![15] The R word "coercion" means values being changed from one type of thing to another type of thing. (We'll ignore the missings and see if they cause us any trouble. The same warning messages will show up on graphs later.) So I have 34 rows (including three rows of missings) instead of the 31 rows I would have liked. Otherwise, success.

There is (inevitably) another way to do this. We are doing the `as.numeric` twice, exactly the same on two different columns, and when you are doing the same thing on a number of columns, here a `mutate` with the same function, you have the option of using `across`. This is the same idea that we used way back to compute numerical summaries of a bunch of columns:

[15]You might think that missing is just missing, but R distinguishes between types of missing.

```r
crickets %>%
  unite(exclamationis, X1:X2) %>%
  unite(niveus, X3:X4) %>%
  pivot_longer(exclamationis:niveus, names_to = "species",
               values_to = "temp_pulse") %>%
  separate(temp_pulse, c("temperature", "pulse_rate"), "_") %>%
  mutate(across(c(temperature, pulse_rate), \(x) as.numeric(x)))
```

```
Warning: There were 2 warnings in `mutate()`.
The first warning was:
i In argument: `across(c(temperature, pulse_rate), function(x) as.numeric(x))`.
Caused by warning:
! NAs introduced by coercion
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```
# A tibble: 34 x 3
   species       temperature pulse_rate
   <chr>               <dbl>      <dbl>
 1 exclamationis        20.8       67.9
 2 niveus               17.2       44.3
 3 exclamationis        20.8       65.1
 4 niveus               18.3       47.2
 5 exclamationis        24         77.3
 6 niveus               18.3       47.6
 7 exclamationis        24         78.7
 8 niveus               18.3       49.6
 9 exclamationis        24         79.4
10 niveus               18.9       50.3
# i 24 more rows
```

Can't I just say that these are columns 2 and 3?

```r
crickets %>%
  unite(exclamationis, X1:X2) %>%
  unite(niveus, X3:X4) %>%
  pivot_longer(exclamationis:niveus, names_to = "species",
               values_to = "temp_pulse") %>%
  separate(temp_pulse, c("temperature", "pulse_rate"), "_") %>%
  mutate(across(2:3, \(x) as.numeric(x)))
```

```
Warning: There were 2 warnings in `mutate()`.
The first warning was:
i In argument: `across(2:3, function(x) as.numeric(x))`.
Caused by warning:
! NAs introduced by coercion
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.

# A tibble: 34 x 3
   species       temperature pulse_rate
   <chr>               <dbl>      <dbl>
 1 exclamationis        20.8       67.9
 2 niveus               17.2       44.3
 3 exclamationis        20.8       65.1
 4 niveus               18.3       47.2
 5 exclamationis        24         77.3
 6 niveus               18.3       47.6
 7 exclamationis        24         78.7
 8 niveus               18.3       49.6
 9 exclamationis        24         79.4
10 niveus               18.9       50.3
# i 24 more rows
```

Yes. Equally good. What goes into the `across` is the same as can go into a `select`: column numbers, names, or any of those "select helpers" like `starts_with`.

You might think of using `across` here on the quantitative columns, but remember the reason for doing this: all the columns are text, before you convert temperature and pulse rate to numbers, and so there's no way to pick out just the two columns you want that way.

Check that the temperature and pulse rate columns are now labelled `dbl`, which means they actually *are* decimal numbers (and don't just look like decimal numbers).

Either way, using `unite` and then `separate` means that all the columns we created we want to keep (or, all the ones we would have wanted to get rid of have already been gotten rid of).

Now we could actually do some statistics. That we do elsewhere.

■

15.20 Number 1 songs

The data file link contains a lot of information about songs popular in 2000. This dataset is untidy. Our ultimate aim is to answer "which song occupied the #1 position for the largest

number of weeks?". To do that, we will build a pipe that starts from the data frame read in from the URL above, and finishes with an answer to the question. I will take you through this step by step. Each part will involve adding something to the pipe you built previously (possibly after removing a line or two that you used to display the previous result).

(a) Read the data and display what you have.

Solution

```
billboard <- read_csv("http://stat405.had.co.nz/data/billboard.csv")
```

```
Rows: 317 Columns: 83
-- Column specification --------------------------------------------------------
Delimiter: ","
chr   (3): artist.inverted, track, genre
dbl  (66): year, x1st.week, x2nd.week, x3rd.week, x4th.week, x5th.week, x6th...
lgl  (11): x66th.week, x67th.week, x68th.week, x69th.week, x70th.week, x71st...
date  (2): date.entered, date.peaked
time  (1): time

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

There are a *lot* of columns. What does this look like?

```
billboard
```

```
# A tibble: 317 x 83
    year artist.inverted    track time  genre date.entered date.peaked x1st.week
   <dbl> <chr>              <chr> <tim> <chr> <date>       <date>          <dbl>
 1  2000 Destiny's Child    Inde~ 03:38 Rock  2000-09-23   2000-11-18         78
 2  2000 Santana            Mari~ 04:18 Rock  2000-02-12   2000-04-08         15
 3  2000 Savage Garden      I Kn~ 04:07 Rock  1999-10-23   2000-01-29         71
 4  2000 Madonna            Music 03:45 Rock  2000-08-12   2000-09-16         41
 5  2000 Aguilera, Christi~ Come~ 03:38 Rock  2000-08-05   2000-10-14         57
 6  2000 Janet              Does~ 04:17 Rock  2000-06-17   2000-08-26         59
 7  2000 Destiny's Child    Say ~ 04:31 Rock  1999-12-25   2000-03-18         83
 8  2000 Iglesias, Enrique  Be W~ 03:36 Latin 2000-04-01   2000-06-24         63
 9  2000 Sisqo              Inco~ 03:52 Rock  2000-06-24   2000-08-12         77
10  2000 Lonestar           Amaz~ 04:25 Coun~ 1999-06-05   2000-03-04         81
# i 307 more rows
# i 75 more variables: x2nd.week <dbl>, x3rd.week <dbl>, x4th.week <dbl>,
```

```
#   x5th.week <dbl>, x6th.week <dbl>, x7th.week <dbl>, x8th.week <dbl>,
#   x9th.week <dbl>, x10th.week <dbl>, x11th.week <dbl>, x12th.week <dbl>,
#   x13th.week <dbl>, x14th.week <dbl>, x15th.week <dbl>, x16th.week <dbl>,
#   x17th.week <dbl>, x18th.week <dbl>, x19th.week <dbl>, x20th.week <dbl>,
#   x21st.week <dbl>, x22nd.week <dbl>, x23rd.week <dbl>, x24th.week <dbl>, ...
```

On yours, you will definitely see a little arrow top right saying "there are more columns", and you will have to click on it several times to see them all. A lot of the ones on the right will be missing.

■

(b) The columns `x1st.week` through `x76th.week` contain the rank of each song in the Billboard chart in that week, with week 1 being the first week that the song appeared in the chart. Convert all these columns into two: an indication of week, called `week`, and of rank, called `rank`. Most songs appeared in the Billboard chart for a lot less than 76 weeks, so there are missing values, which you want to remove. (I say "indication of week" since this will probably be text at the moment). Display your new data frame. Do you have fewer columns? Why do you have a lot more rows? Explain briefly.

Solution

As is often the case, the first step is `pivot_longer`, to reduce all those columns to something easier to deal with. The columns we want to make longer are the ones ending in "week":

```r
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
```

```
# A tibble: 5,307 x 9
    year artist.inverted track  time  genre date.entered date.peaked week   rank
   <dbl> <chr>           <chr>  <tim> <chr> <date>       <date>      <chr> <dbl>
 1  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x1st~    78
 2  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x2nd~    63
 3  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x3rd~    49
 4  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x4th~    33
 5  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x5th~    23
 6  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x6th~    15
 7  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x7th~     7
 8  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x8th~     5
 9  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x9th~     1
10  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x10t~     1
# i 5,297 more rows
```

The "values" (ranks) have missings in them, which we wanted to get rid of.

There are now only 9 columns, a lot fewer than we started with. This is (I didn't need you to say) because we have collected together all those `week` columns into one (a column called `rank` with an indication of which `week` it came from). The logic of the `pivot_longer` is that all those columns contain ranks (which is what make them the same), but they are ranks from different weeks (which is what makes them different).

What has actually happened is that we have turned "wide" format into "long" format. This is not very insightful, so I would like you to go a bit further in your explanation. The original data frame encodes the rank of each song in each week, and what the `pivot_longer` has done is to make that explicit: in the new data frame, each song's rank in each week appears in *one* row, so that there are as many rows as there are song-week combinations. The original data frame had 317 songs over 76 weeks, so this many:

```
317 * 76
```

```
[1] 24092
```

song-week combinations.

Not every song appeared in the Billboard chart for 76 weeks, so our tidy data frame has a lot fewer rows than this.

You need to say that the original data frame had each song appearing once (on one line), but now each song appears on multiple rows, one for each week that the song was in the chart. Or something equivalent to that.

■

(c) Both your `week` and `rank` columns are (probably) text. Create new columns that contain just the numeric values, and display just your new columns, again adding onto the end of your pipe. If it so happens that `rank` is already a number, leave it as it is.

Solution

My `rank` is already a number, so I could leave it; for later, I make a copy of it called `rakn_numbe`. The `week` has a number *in* it, which I can extract using `parse_number`:

```
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
  mutate(week_number=parse_number(week),
         rank_number=rank)
```

```
# A tibble: 5,307 x 11
    year artist.inverted track  time  genre date.entered date.peaked week   rank
   <dbl> <chr>           <chr>  <tim> <chr> <date>       <date>      <chr> <dbl>
 1  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x1st~    78
 2  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x2nd~    63
 3  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x3rd~    49
 4  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x4th~    33
 5  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x5th~    23
 6  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x6th~    15
 7  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x7th~     7
 8  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x8th~     5
 9  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x9th~     1
10  2000 Destiny's Child Indep~ 03:38 Rock  2000-09-23   2000-11-18  x10t~     1
# i 5,297 more rows
# i 2 more variables: week_number <dbl>, rank_number <dbl>
```

■

(e) The meaning of your week-number column is that it refers to the number of weeks *after* the song first appeared in the Billboard chart. That is, if a song's first appearance (in `date.entered`) is July 24, then week 1 is July 24, week 2 is July 31, week 3 is August 7, and so on. Create a column `current` by adding the appropriate number of *days*, based on your week number, to `date.entered`. Display `date.entered`, your week number, and `current` to show that you have calculated the right thing. Note that you can add a number of days onto a date and you will get another date.

Solution

There is a (small) gotcha here: if you read carefully, you'll see that "week 1" is actually "week 0" in terms of the number of days to add on to `date.entered`. So you have to subtract one from the number of weeks before you multiply it by seven to get a number of days. After that thinking, this:

```
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
  mutate(week_number=parse_number(week),
         rank_number=rank) %>%
  mutate(current = date.entered + (week_number - 1) * 7) %>%
  select(date.entered, week_number, current)
```

```
# A tibble: 5,307 x 3
   date.entered week_number current
   <date>             <dbl> <date>
```

```
 1 2000-09-23              1 2000-09-23
 2 2000-09-23              2 2000-09-30
 3 2000-09-23              3 2000-10-07
 4 2000-09-23              4 2000-10-14
 5 2000-09-23              5 2000-10-21
 6 2000-09-23              6 2000-10-28
 7 2000-09-23              7 2000-11-04
 8 2000-09-23              8 2000-11-11
 9 2000-09-23              9 2000-11-18
10 2000-09-23             10 2000-11-25
# i 5,297 more rows
```

Don't forget to use your `week`-turned-into-number, or else it won't work! (This bit me too, so you don't need to feel bad.)

You can also combine the three column-definition statements into one mutate. It doesn't matter; as soon as you have defined a column, you can use it in defining another column, even within the same `mutate`.

Anyway, the rows displayed are all `week_number` 1, so the `current` date should be the same as `date.entered`, and is. (These are all the first week that a song is in the Billboard chart).

You might be thinking that this is not much of a check, and you would be right. A handy trick is to display a random sample of 10 (say) out of the 5,000-odd rows of the data frame. To do that, add the line `sample_n(10)` on the end, like this:

```r
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
  mutate(
    week_number = parse_number(week),
    rank_number = rank
  ) %>%
  mutate(current = date.entered + (week_number - 1) * 7) %>%
  select(date.entered, week_number, current) %>%
  sample_n(10)
```

```
# A tibble: 10 x 3
   date.entered week_number current
   <date>             <dbl> <date>
 1 2000-09-09             7 2000-10-21
 2 2000-11-25            18 2001-03-24
 3 1999-11-13            18 2000-03-11
 4 1999-12-25             1 1999-12-25
```

```
 5 2000-07-08            20 2000-11-18
 6 2000-08-19            13 2000-11-11
 7 2000-08-12            15 2000-11-18
 8 2000-10-21            10 2000-12-23
 9 2000-08-12             2 2000-08-19
10 2000-08-12            19 2000-12-16
```

This gives a variety of rows to check. The first `current` should be $7 - 1 = 6$ weeks, or about a month and a half, after the date the song entered the chart, and so it is; the second and third ones should be $18 - 1 = 17$ weeks after entry, which is very close to a third of a year ($17 \times 3 = 51$), or four months. November to March is indeed four months. The fourth one is the first week on the charts, so the current date and the date entered should be (and are) the same. And so on.

Your random selection of rows is likely to be different from mine, but the same kind of thinking will enable you to check whether it makes sense.

■

(f) Reaching the #1 rank on the Billboard chart is one of the highest accolades in the popular music world. List all the songs that reached `rank` 1. For these songs, list the artist (as given in the data set), the song title, and the date(s) for which the song was ranked number 1. Arrange the songs in date order of being ranked #1. Display all the songs (I found 55 of them).

Solution

To the previous pipe, add the last lines below. You can use either `rank` (text) or what I called `rank_number` (a number). It doesn't matter here, since we are only checking for equal-to, not something like "less than":

```
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
  mutate(week_number=parse_number(week),
         rank_number=rank) %>%
  mutate(current = date.entered + (week_number - 1) * 7) %>%
  filter(rank == 1) %>%
  arrange(current) %>%
  select(artist.inverted, track, current)
```

```
# A tibble: 55 x 3
   artist.inverted     track                 current
   <chr>               <chr>                 <date>
 1 Aguilera, Christina What A Girl Wants     2000-01-15
```

```
 2 Aguilera, Christina What A Girl Wants     2000-01-22
 3 Savage Garden       I Knew I Loved You    2000-01-29
 4 Savage Garden       I Knew I Loved You    2000-02-05
 5 Savage Garden       I Knew I Loved You    2000-02-12
 6 Carey, Mariah       Thank God I Found You 2000-02-19
 7 Savage Garden       I Knew I Loved You    2000-02-26
 8 Lonestar            Amazed                2000-03-04
 9 Lonestar            Amazed                2000-03-11
10 Destiny's Child     Say My Name           2000-03-18
# i 45 more rows
```

You'll see the first ten rows, as here, but with clickable buttons to see the next 10 (and the previous 10 if you have moved beyond 1–10). The "artist" column is called `artist.inverted` because, if the artist is a single person rather than a group, their last name is listed first. The song title appears in the column `track`.

The song by Destiny's Child spills into 2001 because it entered the chart in 2000, and the data set keeps a record of all such songs until they drop out of the chart. I'm not sure what happened to the song that was #1 on January 8, 2000; maybe it entered the chart in 1999[16] and so is not listed here.

■

(g) Use R to find out which song held the #1 rank for the largest number of weeks. For this, you can assume that the song titles are all unique (if it's the same song title, it's the same song), but the artists might not be (for example, Madonna might have had two different songs reach the #1 rank). The information you need is in the output you obtained for the previous part, so it's a matter of adding some code to the end of that. The last mark was for displaying *only* the song that was ranked #1 for the largest number of weeks, or for otherwise making it easy to see which song it was.

Solution

This is a question of using `count`, but on the `track` title:

```
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
  mutate(week_number=parse_number(week),
         rank_number=rank) %>%
  mutate(current = date.entered + (week_number - 1) * 7) %>%
  filter(rank == 1) %>%
  arrange(current) %>%
  select(artist.inverted, track, current) %>%
```

[16]Which was the title of a song by Prince.

```
  count(track)
```

```
# A tibble: 17 x 2
   track                                     n
   <chr>                                 <int>
 1 Amazed                                    2
 2 Be With You                               3
 3 Bent                                      1
 4 Come On Over Baby (All I Want Is You)     4
 5 Doesn't Really Matter                     3
 6 Everything You Want                       1
 7 I Knew I Loved You                        4
 8 Incomplete                                2
 9 Independent Women Part I                 11
10 It's Gonna Be Me                          2
11 Maria, Maria                             10
12 Music                                     4
13 Say My Name                               3
14 Thank God I Found You                     1
15 Try Again                                 1
16 What A Girl Wants                         2
17 With Arms Wide Open                       1
```

Then you can scan down the `n` column, find that the biggest number is 11, and say: it's the song "Independent Women Part I" by Destiny's Child. This is 3 points (out of 4, when the question was to be handed in).

But, this is a data frame, so anything we can do to a data frame we can do to this, like listing out only the row(s) where `n` is equal to its maximum value:

```
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
  mutate(week_number=parse_number(week),
         rank_number=rank) %>%
  mutate(current = date.entered + (week_number - 1) * 7) %>%
  filter(rank == 1) %>%
  arrange(current) %>%
  select(artist.inverted, track, current) %>%
  count(track) %>%
  filter(n == max(n))
```

```
# A tibble: 1 x 2
  track                        n
  <chr>                    <int>
1 Independent Women Part I    11
```

or arranging them in (most logically, descending) order by `n` to make it easier to pick out the top one:

```
billboard %>%
  pivot_longer(ends_with("week"), names_to = "week", values_to="rank", values_drop_na = T)
  mutate(week_number=parse_number(week),
         rank_number=rank) %>%
  mutate(current = date.entered + (week_number - 1) * 7) %>%
  filter(rank == 1) %>%
  arrange(current) %>%
  select(artist.inverted, track, current) %>%
  count(track) %>%
  arrange(desc(n))
```

```
# A tibble: 17 x 2
   track                                     n
   <chr>                                 <int>
 1 Independent Women Part I                 11
 2 Maria, Maria                             10
 3 Come On Over Baby (All I Want Is You)     4
 4 I Knew I Loved You                        4
 5 Music                                     4
 6 Be With You                               3
 7 Doesn't Really Matter                     3
 8 Say My Name                               3
 9 Amazed                                    2
10 Incomplete                                2
11 It's Gonna Be Me                          2
12 What A Girl Wants                         2
13 Bent                                      1
14 Everything You Want                       1
15 Thank God I Found You                     1
16 Try Again                                 1
17 With Arms Wide Open                       1
```

Either of those would have netted you the 4th point.

If you want to be a little bit more careful, you can make an artist-track combination as below. This would catch occasions where the same song by two different artists made it to #1, or two different songs that happened to have the same title did. It's not very likely that the same artist would record two *different* songs with the same title, though it is possible that the same song by the same artist could appear in the Billboard chart on two different occasions.[17]

I think I want to create an artist-song combo fairly early in my pipe, and then display *that* later, something like this. This means replacing `track` by my `combo` later in the pipe, wherever it appears:

```
billboard %>%
  pivot_longer(x1st.week:x76th.week, names_to = "week", values_to = "rank", values_drop_na
  mutate(
    week_number = parse_number(week),
    rank_number = rank
  ) %>%
  mutate(combo = paste(track, artist.inverted, sep = " by ")) %>%
  mutate(current = date.entered + (week_number - 1) * 7) %>%
  filter(rank == 1) %>%
  arrange(current) %>%
  select(combo, current) %>%
  count(combo) %>%
  arrange(desc(n))
```

```
# A tibble: 17 x 2
   combo                                                          n
   <chr>                                                        <int>
 1 Independent Women Part I by Destiny's Child                     11
 2 Maria, Maria by Santana                                         10
 3 Come On Over Baby (All I Want Is You) by Aguilera, Christina     4
 4 I Knew I Loved You by Savage Garden                              4
 5 Music by Madonna                                                 4
 6 Be With You by Iglesias, Enrique                                 3
 7 Doesn't Really Matter by Janet                                   3
 8 Say My Name by Destiny's Child                                   3
 9 Amazed by Lonestar                                               2
10 Incomplete by Sisqo                                              2
11 It's Gonna Be Me by N'Sync                                       2
12 What A Girl Wants by Aguilera, Christina                         2
13 Bent by matchbox twenty                                          1
14 Everything You Want by Vertical Horizon                          1
```

[17] As, for example, when Prince died.

```
15 Thank God I Found You by Carey, Mariah                          1
16 Try Again by Aaliyah                                            1
17 With Arms Wide Open by Creed                                    1
```

I don't think it makes any difference here, but it might in other years, or if you look over several years where you might get cover versions of the same song performed by different artists.

Zero-point bonus: how many of these artists have you heard of? How many have your parents heard of? (I followed popular music quite closely much earlier than this, in the early 1980s in the UK. I remember both Madonna and U2 when they *first* became famous. U2's first single was called "Fire" and it just scraped into the UK top 40. Things changed after that.)

■

15.21 Bikes on College

The City of Toronto collects all kinds of data on aspects of life in the city. See link. One collection of data is records of the number of cyclists on certain downtown streets. The data in link are a record of the cyclists on College Street on the block west from Huron to Spadina on September 24, 2010. In the spreadsheet, each row relates to one cyclist. The first column is the time the cyclist was observed (to the nearest 15 minutes). After that, there are four pairs of columns. The observer filled in (exactly) one X in each pair of columns, according to whether (i) the cyclist was male or female, (ii) was or was not wearing a helmet, (iii) was or was not carrying a passenger on the bike, (iv) was or was not riding on the sidewalk. We want to create a tidy data frame that has the time in each row, and has columns containing appropriate values, often `TRUE` or `FALSE`, for each of the four variables measured.

I will lead you through the process, which will involve developing a (long) pipe, one step at a time.

(a) Take a look at the spreadsheet (using Excel or similar: this may open when you click the link). Are there any obvious header rows? Is there any extra material before the data start? Explain briefly.

Solution

This is what I see (you should see something that looks like this):

	A	B	C	D	E	F	G	H	I	J
1	Ontario Traffic Inc - Bicycle Counts									
2										
3	Location:	College St from Spadina Ave to Huron St				Date:	09/24/2010			
4										
5	Direction:	WB				Weather:	Clear			
6										
7	Start Time	Gender		Wearing Helmet		Passenger On The Bike		Riding On The Sidewalk		
8	(every 15 min)	Male	Female	Yes	No	Yes	No	Yes	No	
9	7:00	X			X		X		X	
10		X			X		X		X	
11		X			X		X		X	
12		X			X		X		X	
13		X			X		X		X	
14	7:15	X		X			X		X	
15			X	X			X		X	
16		X		X			X		X	
17			X	X			X	X		

There are really *two* rows of headers (the rows highlighted in yellow). The actual information that says what the column pair is about is in the first of those two rows, and the second row indicates which category of the information above this column refers to. This is not the usual way that the column headers encode what the columns are about: we are used to having *one* column `gender` that would take the values `female` or `male`, or a column `helmet` containing the values `yes` or `no`. (You might be sensing `pivot_longer` here, which may be one way of tackling this, but I lead you into another idea below.) There are also six lines above the highlighted ones that contain background information about this study. (This is where I got the information about the date of the study and which block of which street it is about.) I am looking for two things: the apparent header line is actually two lines (the ones in yellow), and there are extra lines above that which are not data.

■

(b) Read the data into an R data frame. Read *without* headers, and instruct R how many lines to skip over using `skip=` and a suitable number. When this is working, display the first few lines of your data frame. Note that your columns have names `X1` through `X9`.

Solution

The actual data start on line 9, so we need to skip 8 lines. `col_names=F` is the way to say that we have no column names (not ones that we want to use, anyway). Just typing the name of the data frame will display "a few" (that is, 10) lines of it, so that you can check it for plausibleness:

```
my_url <- "http://ritsokiguess.site/datafiles/bikes.csv"
bikes <- read_csv(my_url, skip = 8, col_names = FALSE)
```

```
Rows: 1958 Columns: 9
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (8): X2, X3, X4, X5, X6, X7, X8, X9
time (1): X1

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bikes
```

```
# A tibble: 1,958 x 9
   X1     X2    X3    X4    X5    X6    X7    X8    X9
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 2    NA  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 3    NA  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 4    NA  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 5    NA  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X
 7    NA  <NA>  X     X     <NA>  <NA>  X     <NA>  X
 8    NA  X     <NA>  X     <NA>  <NA>  X     <NA>  X
 9    NA  <NA>  X     X     <NA>  <NA>  X     X     <NA>
10    NA  X     <NA>  X     <NA>  <NA>  X     <NA>  X
# i 1,948 more rows
```

This seems to have worked: a column with times in it, and four pairs of columns, with exactly one of each pair having an X in it. The variable names `X1` through `X9` were generated by `read_csv`, as it does when you read in data with `col_names=FALSE`. The times are correctly `time`s, and the other columns are all text. The blank cells in the spreadsheet have appeared in our data frame as "missing" (`NA`). The notation `<NA>` means "missing text" (as opposed to a missing number, say).

The first line in our data frame contains the first 7:00 (am) cyclist, so it looks as if we skipped the right number of lines.

■

(c) What do you notice about the times in your first column? What do you think those "missing" times should be?

Solution

There are some times and some missing values. It seems a reasonable guess that the person recording the data only recorded a time when a new period of 15 minutes had begun, so that the missing times should be the same as the previous non-missing one: For example, the first five rows are cyclists observed at 7:00 am (or, at least, between 7:00 and 7:15). So they should be recorded as 7:00, and the ones in rows 7–10 should be recorded as 7:15, and so on.

■

(d) Find something from the `tidyverse` that will fill[18] in those missing values with the right thing. Start a pipe from the data frame you read in, that updates the appropriate column with the filled-in times.

Solution

`fill` from `tidyr` fills in the missing times with the previous non-missing value. (This will mean finding the help for `fill` in R Studio or online.) I told you it was a giveaway. If you look in the help for `fill` via `?fill` (or if you Google `tidyr::fill`, which is the full name for "the `fill` that lives in `tidyr`"), you'll see that it requires up to two things (not including the data frame): a column to fill, and a direction to fill it (the default of "down" is exactly what we want). Thus:

```
bikes %>% fill(X1)
```

```
# A tibble: 1,958 x 9
   X1     X2    X3    X4    X5    X6    X7    X8    X9
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 2 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 3 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 4 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 5 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X
 7 07:15  <NA>  X     X     <NA>  <NA>  X     <NA>  X
 8 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X
 9 07:15  <NA>  X     X     <NA>  <NA>  X     X     <NA>
10 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X
# i 1,948 more rows
```

Success!

[18] Oh, what a giveaway.

We will probably want to rename `X1` to something like `time`, so let's do that now before we forget. There is a `rename` that does about what you'd expect:

```
bikes %>% fill(X1) %>% rename(Time = X1)
```

```
# A tibble: 1,958 x 9
   Time   X2    X3    X4    X5    X6    X7    X8    X9
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 2 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 3 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 4 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 5 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X
 7 07:15  <NA>  X     X     <NA>  <NA>  X     <NA>  X
 8 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X
 9 07:15  <NA>  X     X     <NA>  <NA>  X     X     <NA>
10 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X
# i 1,948 more rows
```

The only thing I keep forgetting is that the syntax of `rename` is "new name equals old name". Sometimes I think it's the other way around, and then I wonder why it doesn't work.

I gave it a capital T so as not to confuse it with other things in R called `time`.

■

(e) R's `ifelse` function works like `=IF` in Excel. You use it to create values for a new variable, for example in a `mutate`. The first input to it is a logical condition (something that is either true or false); the second is the value your new variable should take if the condition is true, and the third is the value of your new variable if the condition is false. Create a new column `gender` in your data frame that is "male" or "female" depending on the value of your `X2` column, using `mutate`. (You can assume that exactly one of the second and third columns has an `X` in it.) Add your code to the end of your pipe and display (the first 10 rows of) the result.

Solution

Under the assumption we are making, we only have to look at column `X2` and we ignore `X3` totally:

```
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
```

```
    mutate(gender = ifelse(X2 == "X", "male", "female"))
```

```
# A tibble: 1,958 x 10
   Time   X2    X3    X4    X5    X6    X7    X8    X9    gender
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 2 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 3 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 4 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 5 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male
 7 07:15  <NA>  X     X     <NA>  <NA>  X     <NA>  X     <NA>
 8 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male
 9 07:15  <NA>  X     X     <NA>  <NA>  X     X     <NA>  <NA>
10 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male
# i 1,948 more rows
```

Oh, that didn't work. The gender column is either `male` or missing; the two missing ones here should say `female`. What happened? Let's just look at our logical condition this time:

```
  bikes %>%
    fill(X1) %>%
    rename(Time = X1) %>%
    mutate(isX = (X2 == "X"))
```

```
# A tibble: 1,958 x 10
   Time   X2    X3    X4    X5    X6    X7    X8    X9    isX
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <lgl>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     TRUE
 2 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     TRUE
 3 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     TRUE
 4 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     TRUE
 5 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     TRUE
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     TRUE
 7 07:15  <NA>  X     X     <NA>  <NA>  X     <NA>  X     NA
 8 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     TRUE
 9 07:15  <NA>  X     X     <NA>  <NA>  X     X     <NA>  NA
10 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     TRUE
# i 1,948 more rows
```

This is not true and false, it is true and missing. The idea is that if `X2` is missing, we don't (in general) know what its value is: it might even be `X`! So if `X2` is missing, any comparison of it with another value ought to be missing as well.

That's in general. Here, we know where those missing values came from: they were blank cells in the spreadsheet, so we actually have more information.

Perhaps a better way to go is to test whether `X2` is missing (in which case, it's a female cyclist). R has a function `is.na` which is `TRUE` if the thing inside it is missing and `FALSE` if the thing inside it has some non-missing value. In our case, it goes like this:

```
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = ifelse(is.na(X2), "female", "male"))
```

```
# A tibble: 1,958 x 10
   Time   X2    X3    X4    X5    X6    X7    X8    X9    gender
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 2 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 3 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 4 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 5 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male
 7 07:15  <NA>  X     X     <NA>  <NA>  X     <NA>  X     female
 8 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male
 9 07:15  <NA>  X     X     <NA>  <NA>  X     X     <NA>  female
10 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male
# i 1,948 more rows
```

Or you can test `X3` for missingness: if missing, it's male, otherwise it's female. That also works.

This made an assumption that the person recording the X's actually *did* mark an X in exactly one of the columns. For example, the columns could *both* be missing, or *both* have an X in them. This gives us more things to check, at least three. `ifelse` is good for something with only two alternatives, but when you have more, `case_when` is much better.[19] Here's how that goes. Our strategy is to check for three things: (i) `X2` has an `X` and `X3` is missing; (ii) `X2` is missing and `X3` has an `X`; (iii) anything else, which is an error:

[19] In some languages it is called `switch`. Python appears not to have it. What you do there instead is to use a Python dictionary to pick out the value you want.

```r
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = case_when(
    X2 == "X" & is.na(X3) ~ "Male",
    is.na(X2) & X3 == "X" ~ "Female",
    TRUE                  ~ "Error!"
  ))
```

```
# A tibble: 1,958 x 10
   Time   X2    X3    X4    X5    X6    X7    X8    X9    gender
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     Male
 2 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     Male
 3 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     Male
 4 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     Male
 5 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     Male
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     Male
 7 07:15  <NA>  X     X     <NA>  <NA>  X     <NA>  X     Female
 8 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     Male
 9 07:15  <NA>  X     X     <NA>  <NA>  X     X     <NA>  Female
10 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     Male
# i 1,948 more rows
```

It seems nicest to format it with the squiggles lining up, so you can see what possible values `gender` might take.

The structure of the `case_when` is that the thing you're checking for goes on the left of the squiggle, and the value you want your new variable to take goes on the right. What it does is to go down the list of conditions that you are checking for, and as soon as it finds one that is true, it grabs the value on the right of the squiggle and moves on to the next row. The usual way to write these is to have a catch-all condition at the end that is always true, serving to make sure that your new variable always gets *some* value. `TRUE` is, um, always true. If you want an English word for the last condition of your `case_when`, "otherwise" is a nice one.

I wanted to check that the observer did check exactly one of `V2` and `V3` as I asserted, which can be done by gluing this onto the end:

```r
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = case_when(
```

```
      X2 == "X" & is.na(X3) ~ "Male",
      is.na(X2) & X3 == "X" ~ "Female",
      TRUE ~ "Error!"
    )) %>%
    count(gender)
```

```
# A tibble: 2 x 2
  gender     n
  <chr>  <int>
1 Female   861
2 Male    1097
```

There are only Males and Females, so the observer really did mark exactly one X. (As a bonus, you see that there were slightly more male cyclists than female ones.)

Extra: I was wondering how `pivot_longer` would play out here. The way to do it seems to be to rename the columns we want first, and get rid of the others:

```
  bikes %>%
    fill(X1) %>%
    rename(Time = X1) %>%
    rename(male = X2, female = X3) %>%
    select(-starts_with("X"))
```

```
# A tibble: 1,958 x 3
   Time   male  female
   <time> <chr> <chr>
 1 07:00  X     <NA>
 2 07:00  X     <NA>
 3 07:00  X     <NA>
 4 07:00  X     <NA>
 5 07:00  X     <NA>
 6 07:15  X     <NA>
 7 07:15  <NA>  X
 8 07:15  X     <NA>
 9 07:15  <NA>  X
10 07:15  X     <NA>
# i 1,948 more rows
```

Each row should have one X and one missing in it, so we may as well drop the missings as we pivot-longer:

```r
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  rename(male = X2, female = X3) %>%
  select(-starts_with("X")) %>%
  pivot_longer(-Time, names_to="gender", values_to="observed",
               values_drop_na = TRUE)
```

```
# A tibble: 1,958 x 3
   Time   gender observed
   <time> <chr>  <chr>
 1 07:00  male   X
 2 07:00  male   X
 3 07:00  male   X
 4 07:00  male   X
 5 07:00  male   X
 6 07:15  male   X
 7 07:15  female X
 8 07:15  male   X
 9 07:15  female X
10 07:15  male   X
# i 1,948 more rows
```

The `observed` column is kind of pointless, since its value is always X. But we do have a check: the previous data frame had 1958 rows, with an X in either the male or the female column. This data frame has the gender of each observed cyclist in the `gender` column, and it also has 1958 rows. So, either way, that's how many cyclists were observed in total.

■

(f) Create variables `helmet`, `passenger` and `sidewalk` in your data frame that are `TRUE` if the "Yes" column contains `X` and `FALSE` otherwise. This will use `mutate` again, but you don't need `ifelse`: just set the variable equal to the appropriate logical condition. As before, the best way to create these variables is to test the appropriate things for missingness. Note that you can create as many new variables as you like in one `mutate`. Show the first few lines of your new data frame. (Add your code onto the end of the pipe you made above.)

Solution

On the face of it, the way to do this is to go looking for `X`'s:

```r
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = ifelse(is.na(X2), "female", "male")) %>%
  mutate(
    helmet = (X4 == "X"),
    passenger = (X6 == "X"),
    sidewalk = (X8 == "X")
  )
```

```
# A tibble: 1,958 x 13
   Time  X2    X3    X4    X5    X6    X7    X8    X9    gender helmet passenger
   <tim> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>  <lgl>  <lgl>
 1 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   NA     NA
 2 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   NA     NA
 3 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   NA     NA
 4 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   NA     NA
 5 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   NA     NA
 6 07:15 X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE   NA
 7 07:15 <NA>  X     X     <NA>  <NA>  X     <NA>  X     female TRUE   NA
 8 07:15 X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE   NA
 9 07:15 <NA>  X     X     <NA>  <NA>  X     X     <NA>  female TRUE   NA
10 07:15 X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE   NA
# i 1,948 more rows
# i 1 more variable: sidewalk <lgl>
```

But, we run into the same problem that we did with `gender`: the new variables are either `TRUE` or missing, never `FALSE`.

The solution is the same: look for the things that are *missing* if the cyclist is wearing a helmet, carrying a passenger or riding on the sidewalk. These are `X5, X7, X9` respectively:

```r
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = ifelse(is.na(X2), "female", "male")) %>%
  mutate(
    helmet = is.na(X5),
    passenger = is.na(X7),
    sidewalk = is.na(X9)
  )
```

```
# A tibble: 1,958 x 13
   Time  X2    X3    X4    X5    X6    X7    X8    X9    gender helmet passenger
   <tim> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>  <lgl>  <lgl>
 1 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE  FALSE
 2 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE  FALSE
 3 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE  FALSE
 4 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE  FALSE
 5 07:00 X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE  FALSE
 6 07:15 X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE   FALSE
 7 07:15 <NA>  X     X     <NA>  <NA>  X     <NA>  X     female TRUE   FALSE
 8 07:15 X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE   FALSE
 9 07:15 <NA>  X     X     <NA>  <NA>  X     X     <NA>  female TRUE   FALSE
10 07:15 X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE   FALSE
# i 1,948 more rows
# i 1 more variable: sidewalk <lgl>
```

Again, you can do the `mutate` all on one line if you want to, or all four variable assignments in one `mutate`, but I used newlines and indentation to make the structure clear.

It is less elegant, though equally good for the purposes of the assignment, to use `ifelse` for these as well, which would go like this, for example:

```r
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = ifelse(X2 == "X", "male", "female")) %>%
  mutate(helmet = ifelse(is.na(X5), TRUE, FALSE))
```

```
# A tibble: 1,958 x 11
   Time   X2    X3    X4    X5    X6    X7    X8    X9    gender helmet
   <time> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>  <lgl>
 1 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE
 2 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE
 3 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE
 4 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE
 5 07:00  X     <NA>  <NA>  X     <NA>  X     <NA>  X     male   FALSE
 6 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE
 7 07:15  <NA>  X     X     <NA>  <NA>  X     <NA>  X     <NA>   TRUE
 8 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE
 9 07:15  <NA>  X     X     <NA>  <NA>  X     X     <NA>  <NA>   TRUE
10 07:15  X     <NA>  X     <NA>  <NA>  X     <NA>  X     male   TRUE
# i 1,948 more rows
```

and the same for `passenger` and `sidewalk`. The warning is, whenever you see a `TRUE` and a `FALSE` in an `ifelse`, that you could probably get rid of the `ifelse` and use the logical condition directly.[20]

For `gender`, though, you need the `ifelse` (or a `case_when`) because the values you want it to take are `male` and `female`, something other than `TRUE` and `FALSE`.

I like to put brackets around logical conditions when I am assigning them to a variable or defining new columns containing them. If I don't, I get something like

```
helmet <- V4 == "X"
```

which actually works, but is hard to read. Well, I *think* it works. Let's check:

```
exes <- c("X", "", "X", "", "X")
y <- exes == "X"
y
```

```
[1]  TRUE FALSE  TRUE FALSE  TRUE
```

Yes it does. But I would not recommend writing it this way, because unless you are paying attention, you won't notice that `==` is testing for "logically equal" rather than putting something in a column.

It works because of a thing called "operator precedence": the logical-equals is evaluated first, and the result of that is saved in the variable. But unless you or your readers remember that, it's better to write

```
y <- (exes == "X")
```

to draw attention to the order of calculation. This is the same reason that

```
4 + 5 * 6
```

```
[1] 34
```

evaluates this way rather than doing the addition first and getting 54. BODMAS and all that.

The `pivot_longer` approach works for these too. Rename the columns as `yes` and `no`, and then give the `names_to` column a name like `helmet`. Give the `values_to` column a name

[20] If I was helping you, and you were struggling with *ifelse* but finally mastered it, it seemed easier to suggest that you used it again for the others.

like `what2`, to make it easier to remove later. And then do the same with the others, one `pivot_longer` at a time. (Keep all the columns, and then discard them at the end if you want. That way you don't risk deleting something you might need later.)

■

(g) Finally (for the data manipulation), get rid of all the original columns, keeping only the new ones that you created. Save the results in a data frame and display its first few rows.

Solution

This is a breath of fresh air after all the thinking needed above: this is just `select`, added to the end:

```
mybikes <- bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = ifelse(is.na(X2), "female", "male")) %>%
  mutate(
    helmet = is.na(X5),
    passenger = is.na(X7),
    sidewalk = is.na(X9)
  ) %>%
  select(-(X2:X9))
mybikes
```

```
# A tibble: 1,958 x 5
   Time   gender helmet passenger sidewalk
   <time> <chr>  <lgl>  <lgl>     <lgl>
 1 07:00  male   FALSE  FALSE     FALSE
 2 07:00  male   FALSE  FALSE     FALSE
 3 07:00  male   FALSE  FALSE     FALSE
 4 07:00  male   FALSE  FALSE     FALSE
 5 07:00  male   FALSE  FALSE     FALSE
 6 07:15  male   TRUE   FALSE     FALSE
 7 07:15  female TRUE   FALSE     FALSE
 8 07:15  male   TRUE   FALSE     FALSE
 9 07:15  female TRUE   FALSE     TRUE
10 07:15  male   TRUE   FALSE     FALSE
# i 1,948 more rows
```

You might not have renamed your `X1`, in which case, you still have it, but need to keep it (because it holds the times).

Another way to do this is to use a "select-helper", thus:

```
bikes %>%
  fill(X1) %>%
  rename(Time = X1) %>%
  mutate(gender = ifelse(is.na(X2), "female", "male")) %>%
  mutate(
    helmet = is.na(X5),
    passenger = is.na(X7),
    sidewalk = is.na(X9)
  ) %>%
  select(-num_range("X", 2:9))
```

```
# A tibble: 1,958 x 5
   Time   gender helmet passenger sidewalk
   <time> <chr>  <lgl>  <lgl>     <lgl>
 1 07:00  male   FALSE  FALSE     FALSE
 2 07:00  male   FALSE  FALSE     FALSE
 3 07:00  male   FALSE  FALSE     FALSE
 4 07:00  male   FALSE  FALSE     FALSE
 5 07:00  male   FALSE  FALSE     FALSE
 6 07:15  male   TRUE   FALSE     FALSE
 7 07:15  female TRUE   FALSE     FALSE
 8 07:15  male   TRUE   FALSE     FALSE
 9 07:15  female TRUE   FALSE     TRUE
10 07:15  male   TRUE   FALSE     FALSE
# i 1,948 more rows
```

This means "get rid of all the columns whose names are X followed by a number 2 through 9".

The pipe looks long and forbidding, but you built it (and tested it) a little at a time. Which is how you do it.

■

(h) The next few parts are a quick-fire analysis of the data set. They can all be solved using `count`. How many male and how many female cyclists were observed in total?

Solution

I already got this one when I was checking for observer-notation errors earlier:

```
mybikes %>% count(gender)
```

```
# A tibble: 2 x 2
  gender     n
  <chr>  <int>
1 female   861
2 male    1097
```

861 females and 1097 males.

■

(i) How many male and female cyclists were not wearing helmets?

Solution

You can count two variables at once, in which case you get counts of all combinations of them:

```
mybikes %>% count(gender, helmet)
```

```
# A tibble: 4 x 3
  gender helmet     n
  <chr>  <lgl>  <int>
1 female FALSE    403
2 female TRUE     458
3 male   FALSE    604
4 male   TRUE     493
```

403 females and 604 males were not wearing helmets, picking out what we need.

The real question of interest here is "what *proportion* of male and female cyclists were not wearing helmets?"[21] This has a rather elegant solution that I will have to explain. First, let's go back to the `group_by` and `summarize` version of the `count` here:

```
mybikes %>%
  group_by(gender, helmet) %>%
  summarize(the_count = n())
```

[21] But I didn't want to complicate this question any farther.

```
`summarise()` has grouped output by 'gender'. You can override using the
`.groups` argument.

# A tibble: 4 x 3
# Groups:   gender [2]
  gender helmet the_count
  <chr>  <lgl>      <int>
1 female FALSE        403
2 female TRUE         458
3 male   FALSE        604
4 male   TRUE         493
```

That's the same table we got just now. Now, let's calculate a proportion and see what happens:

```
mybikes %>%
  group_by(gender, helmet) %>%
  summarize(the_count = n()) %>%
  mutate(prop = the_count / sum(the_count))
```

```
`summarise()` has grouped output by 'gender'. You can override using the
`.groups` argument.

# A tibble: 4 x 4
# Groups:   gender [2]
  gender helmet the_count  prop
  <chr>  <lgl>      <int> <dbl>
1 female FALSE        403 0.468
2 female TRUE         458 0.532
3 male   FALSE        604 0.551
4 male   TRUE         493 0.449
```

We seem to have the proportions of males and females who were and were not wearing a helmet, and you can check that this is indeed the case, for example:

```
403 / (403 + 458)
```

```
[1] 0.4680604
```

47% of females were not wearing helmets, while 55% of males were helmetless. (You can tell from the original frequencies that a small majority of females wore helmets and a small majority of males did not.)

Now, we have to ask ourselves: how on earth did that work?

When you calculate a summary (like our `sum(count)` above), it figures that you can't want the sum by gender-helmet combination, since you already have those in `count`. You must want the sum *over* something. What? What happens is that it goes back to the `group_by` and "peels off" the last thing there, which in this case is `helmet`, leaving only `gender`. It then sums the counts for each gender, giving us what we wanted.

It just blows my mind that someone (ie., Hadley Wickham) could (i) think that this would be a nice syntax to have (instead of just being an error), (ii) find a way to implement it and (iii) find a nice logical explanation ("peeling off") to explain how it worked.

What happens if we switch the order of the things in the `group_by`?

```
mybikes %>%
  group_by(helmet, gender) %>%
  summarize(the_count = n()) %>%
  mutate(prop = the_count / sum(the_count))
```

```
`summarise()` has grouped output by 'helmet'. You can override using the
`.groups` argument.
```

```
# A tibble: 4 x 4
# Groups:   helmet [2]
  helmet gender the_count  prop
  <lgl>  <chr>      <int> <dbl>
1 FALSE  female       403 0.400
2 FALSE  male         604 0.600
3 TRUE   female       458 0.482
4 TRUE   male         493 0.518
```

Now we get the proportion of helmeted riders of each gender, which is not the same as what we had before. Before, we had "out of males" and "out of females"; now we have "out of helmeted riders" and "out of helmetless riders". (The riders with helmets are almost 50–50 males and females, but the riders without helmets are about 60% male.)

This is row and column proportions in a contingency table, B22 style.

Now, I have to see whether the `count` variant of this works:

```r
mybikes %>%
  count(gender, helmet) %>%
  mutate(prop = n / sum(n))
```

```
# A tibble: 4 x 4
  gender helmet     n  prop
  <chr>  <lgl>  <int> <dbl>
1 female FALSE    403 0.206
2 female TRUE     458 0.234
3 male   FALSE    604 0.308
4 male   TRUE     493 0.252
```

It doesn't. Well, it kind of does, but it divided by the sum of all of them rather than "peeling off", so these are overall proportions rather than row or column proportions.

So I think you have to do this the `group_by` and `summarize` way.

■

(j) How many cyclists were riding on the sidewalk *and* carrying a passenger?

Solution

Not too many, I'd hope. Again:

```r
mybikes %>% count(passenger, sidewalk)
```

```
# A tibble: 3 x 3
  passenger sidewalk     n
  <lgl>     <lgl>    <int>
1 FALSE     FALSE     1880
2 FALSE     TRUE        73
3 TRUE      FALSE        5
```

We're looking for the "true", "true" entry of that table, which seems to have vanished. That means the count is *zero*: none at all. (There were only 5 passenger-carrying riders, and they were all on the road.)

■

(k) What was the busiest 15-minute period of the day, and how many cyclists were there then?

Solution

The obvious way is to list every 15-minute period and eyeball the largest frequency. There are quite a few 15-minute periods, so be prepared to hit Next a few times (or use `View`):

```
mybikes %>% count(Time)
```

```
# A tibble: 48 x 2
   Time       n
   <time> <int>
 1 07:00      5
 2 07:15      8
 3 07:30      9
 4 07:45      5
 5 08:00     18
 6 08:15     14
 7 08:30     12
 8 08:45     22
 9 09:00     17
10 09:15     15
# i 38 more rows
```

17:15, or 5:15 pm, with 128 cyclists.

But, computers are meant to save us that kind of effort. How? Note that the output from `count` is itself a data frame, so anything you can do to a data frame, you can do to *it*: for example, display only the rows where the frequency equals the maximum frequency:

```
mybikes %>%
  count(Time) %>%
  filter(n == max(n))
```

```
# A tibble: 1 x 2
  Time       n
  <time> <int>
1 17:15    128
```

That will actually display *all* the times where the cyclist count equals the maximum, of which there might be more than one.

■

15.22 Feeling the heat

In summer, the city of Toronto issues Heat Alerts for "high heat or humidity that is expected to last two or more days". The precise definitions are shown at link. During a heat alert, the city opens Cooling Centres and may extend the hours of operation of city swimming pools, among other things. All the heat alert days from 2001 to 2016 are listed at link.

The word "warning" is sometimes used in place of "alert" in these data. They mean the same thing.[22]

(a) Read the data into R, and display the data frame. Note that there are four columns:

- a numerical `id` (numbered upwards from the first Heat Alert in 2001; some of the numbers are missing)
- the `date` of the heat alert, in year-month-day format with 4-digit years.
- a text `code` for the type of heat alert
- `text` describing the kind of heat alert. This can be quite long.

Solution

A `.csv`, so:

```
my_url <- "http://ritsokiguess.site/datafiles/heat.csv"
heat <- read_csv(my_url)
```

```
Rows: 200 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (2): code, text
dbl  (1): id
date (1): date

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
heat
```

[22]Unlike *thunderstorm watch* and *thunderstorm warning*, which mean different things.

```
# A tibble: 200 x 4
      id date       code  text
   <dbl> <date>     <chr> <chr>
 1   232 2016-09-08 HAU   Toronto's Medical Officer of Health has upgraded the ~
 2   231 2016-09-07 HAE   Toronto's Medical Officer of Health has continued the~
 3   230 2016-09-06 HA    Toronto's Medical Officer of Health has issued a Heat~
 4   228 2016-08-13 EHAE  Toronto's Medical Officer of Health has continued the~
 5   227 2016-08-12 EHAE  Toronto's Medical Officer of Health has continued the~
 6   226 2016-08-11 HAU   Toronto's Medical Officer of Health has upgraded the ~
 7   225 2016-08-10 HAE   Toronto's Medical Officer of Health has continued the~
 8   224 2016-08-09 HA    Toronto's Medical Officer of Health has issued a Heat~
 9   222 2016-08-05 HAE   Toronto's Medical Officer of Health has continued the~
10   221 2016-08-04 HA    Toronto's Medical Officer of Health has issued a Heat~
# i 190 more rows
```

You might get a truncated `text` as I did, or you might have to click to see more of it. In any case, we won't be using the text, so you can just forget about it from here on.

■

(b) In your data frame, are the dates stored as genuine dates or as text? How can you tell?

Solution

Look at the top of the column on your display of the data frame. Under the date column it says `date` rather than `chr` (which means "text"), so these are genuine dates. This happened because the data file contained the dates in year-month-day order, so `read_csv` read them in as dates. (If they had been in some other order, they would have been read in as text and we would need to use `lubridate` to make them into dates.)

■

(c) Which different heat alert codes do you have, and how many of each?

Solution

`count`, most easily:

```
heat %>% count(code)
```

```
# A tibble: 6 x 2
  code      n
  <chr> <int>
1 EHA      59
2 EHAD      1
```

```
3 EHAE     18
4 HA       93
5 HAE      16
6 HAU      13
```

Alternatively, `group_by` and `summarize`:

```
heat %>% group_by(code) %>% summarize(count = n())
```

```
# A tibble: 6 x 2
  code  count
  <chr> <int>
1 EHA      59
2 EHAD      1
3 EHAE     18
4 HA       93
5 HAE      16
6 HAU      13
```

(note that `n()` gives the number of rows, in each group if you have groups.)

There are six different codes, but EHAD only appears once.

■

(d) Use the `text` in your dataset (or look back at the original data file) to describe briefly in your own words what the various codes represent.

Solution

You can check that each time a certain code appears, the text next to it is identical. The six codes and my brief descriptions are:

- EHA: (Start of) Extended Heat Alert
- EHAD: Extreme Heat Alert downgraded to Heat Alert
- EHAE: Extended Heat Alert continues
- HA: (Start of) Heat Alert
- HAE: Heat Alert continues
- HAU: Heat Alert upgraded to Extended Heat Alert

I thought there was such a thing as an Extreme Heat Alert, but here the word is (usually) Extended, meaning a heat alert that extends over several days, long in duration rather than extremely hot. The only place Extreme occurs is in EHAD, which only occurs once. I want

your answer to say or suggest something about whether a code applies *only* to continuing heat alerts (ie., that EHAD, EHAE, HAE and HAU are different from the others).

■

(e) How many (regular and extended) heat alert events are there altogether? A heat alert event is a stretch of consecutive days, on all of which there is a heat alert or extended heat alert. Hints: (i) you can answer this from output you already have; (ii) how can you tell when a heat alert event *starts*?

Solution

This turned out to be more messed-up than I thought. There is a detailed discussion below. The codes EHAD, EHAE, HAE, HAU all indicate that there was a heat alert on the day before. Only the codes HA and EHA can indicate the start of a heat alert (event). The problem is that HA and EHA sometimes indicate the start of a heat alert event and sometimes one that is continuing. You can check by looking at the data that HA and EHA days can (though they don't always: see below) have a non-heat-alert day before (below) them in the data file: for example, August 4, 2012 is an HA day, but August 3 of that year was not part of any kind of heat alert. I had intended the answer to be this:

> So we get the total number of heat alert events by totalling up the number of HA and EHA days: $59 + 93 = 152$.

This is not right because there are some consecutive EHA days, eg. 5–8 July 2010, so that EHA sometimes indicates the continuation of an extended heat alert and sometimes the start of one. I was expecting EHA to be used only for the start, and one of the other codes to indicate a continuation. The same is (sometimes) true of HA. So reasonable answers to the question as set include:

- 93, the number of HAs
- 59, the number of EHAs
- 152, the number of HAs and EHAs combined
- "152 or less", "between 93 and 152", "between 59 and 152' ' to reflect that not all of these mark the start of a heat alert event.

Any of these, or something similar *with an explanation of how you got your answer*, are acceptable. In your career as a data scientist, you will often run into this kind of thing, and it will be your job to do something with the data and *explain* what you did so that somebody else can decide whether they believe you or not. A good explanation, even if it is not correct, will help you get at the truth because it will inspire someone to say "in fact, it goes *this* way", and then the two of you can jointly figure out what's actually going on. Detailed discussion follows. If you have *any* ambitions of working with data, you should try to follow the paragraphs below, because they indicate how you would get an *actual* answer to the question.

I think the key is the number of days between one heat alert day and the next one. `dplyr` has a function `diff` that works out exactly this. Building a pipeline, just because:

```
heat %>%
  select(-text) %>%
  mutate(daycount = as.numeric(date)) %>%
  mutate(daydiff = abs(c(diff(daycount), 0)))
```

```
# A tibble: 200 x 5
      id date       code  daycount daydiff
   <dbl> <date>     <chr>    <dbl>   <dbl>
 1   232 2016-09-08 HAU      17052       1
 2   231 2016-09-07 HAE      17051       1
 3   230 2016-09-06 HA       17050      24
 4   228 2016-08-13 EHAE     17026       1
 5   227 2016-08-12 EHAE     17025       1
 6   226 2016-08-11 HAU      17024       1
 7   225 2016-08-10 HAE      17023       1
 8   224 2016-08-09 HA       17022       4
 9   222 2016-08-05 HAE      17018       1
10   221 2016-08-04 HA       17017      11
# i 190 more rows
```

Oof. I have some things to keep track of here:

- Get rid of the `text`, since it serves no purpose here.
- The `date` column is a proper `Date` (we checked).
- Then I want the date as number of days; since it is a number of days internally, I just make it a number with `as.numeric`.
- Then I use `diff` to get the difference between each date and the previous one, remembering to glue a 0 onto the end so that I have the right number of differences.
- Since the dates are most recent first, I take the absolute value so that the `daydiff` values are positive (except for the one that is 0 on the end).

Still with me? All right. You can check that the `daydiff` values are the number of days between the date on that line and the line below it. For example, there were 24 days between August 13 and September 6.

Now, when `daydiff` is 1, there was also a heat alert on the previous day (the line below in the file), but when `daydiff` is *not* 1, that day must have been the *start* of a heat alert event.

So if I count the non-1's, that will count the number of heat alert events there were. (That includes the difference of 0 on the first day, the one at the end of the file.)

Thus my pipeline continues like this:

```
heat %>%
  select(-text) %>%
  mutate(daycount = as.numeric(date)) %>%
  mutate(daydiff = abs(c(diff(daycount), 0))) %>%
  count(daydiff != 1)
```

```
# A tibble: 2 x 2
  `daydiff != 1`     n
  <lgl>          <int>
1 FALSE            121
2 TRUE              79
```

And that's how many actual heat alert events there were: 79, less even than the number of HAs. So that tells me that a lot of my HAs and EHAs were actually continuations of heat alert events rather than the start of them. I think I need to have a word with the City of Toronto about their data collection processes.

`count` will count anything that is, or can be made into, a categorical variable. It doesn't have to be one of the columns of your data frame; here it is something that is either `TRUE` or `FALSE` about every row of the data frame.

One step further: what *is* the connection between the codes and the start of heat alert events? We can figure that out now:

```
heat %>%
  select(-text) %>%
  mutate(daycount = as.numeric(date)) %>%
  mutate(daydiff = abs(c(diff(daycount), 0))) %>%
  mutate(start = (daydiff != 1)) %>%
  count(code, start)
```

```
# A tibble: 8 x 3
  code  start     n
  <chr> <lgl> <int>
1 EHA   FALSE    53
2 EHA   TRUE      6
3 EHAD  FALSE     1
```

```
4 EHAE  FALSE    18
5 HA    FALSE    20
6 HA    TRUE     73
7 HAE   FALSE    16
8 HAU   FALSE    13
```

I made a column `start` that is `TRUE` at the start of a heat alert event and `FALSE` otherwise, by comparing the days from the previous heat alert day with 1. Then I can make a `table`, or, as here, the `dplyr` equivalent with `count`.[23] Or `group_by` and `summarize`. What this shows is that EHAD, EHAE, HAE and HAU *never* go with the start of a heat alert event (as they shouldn't). But look at the HAs and EHAs. For the HAs, 73 of them go with the start of an event, but 20 do not. For the EHAs, just 6 of them go with the start, and 53 do not. (Thus, counting just the HAs was very much a reasonable thing to do.)

The 79 heat alert events that we found above had 73 of them starting with an HA, and just 6 starting with an EHA. I wasn't quite sure how this would come out, but I knew it had something to do with the number of days between one heat alert day and the next, so I calculated those first and then figured out what to do with them.

■

(f) We are going to investigate how many heat alert days there were in each year. To do that, we have to extract the year from each of our dates.

Solution

This will need the `lubridate` package, but you don't need to load it specifically because it is now loaded with the `tidyverse`:

```
heat %>% select(-text) %>% mutate(year = year(date)) %>% sample_n(10)
```

```
# A tibble: 10 x 4
      id date       code   year
   <dbl> <date>     <chr> <dbl>
 1   219 2016-07-24 EHAE   2016
 2    47 2005-07-10 HA     2005
 3    67 2006-07-03 EHA    2006
 4   176 2013-06-26 EHAD   2013
 5   193 2015-07-28 HAE    2015
 6   141 2012-06-19 HA     2012
 7    55 2005-07-18 EHA    2005
```

[23] I did not know until just now that you could put two variables in a count and you get counts of all the combinations of them. Just goes to show the value of "try it and see".

```
 8   119 2010-09-02 EHA    2010
 9   102 2009-06-25 EHA    2009
10     9 2001-08-09 EHA    2001
```

That seems to have worked. I listed a random sample of rows to get back to previous years. Having convinced myself that it worked, let me save it:

```
heat %>% select(-text) %>% mutate(year = year(date)) -> heat
```

■

(g) Count the number of heat alert days for each year, by tabulating the year variable. Looking at this table, would you say that there have been more heat alert days in recent years? Explain (very) briefly.

Solution

Count them again:

```
heat %>% count(year)
```

```
# A tibble: 16 x 2
    year     n
   <dbl> <int>
 1  2001     9
 2  2002    16
 3  2003     6
 4  2004     2
 5  2005    26
 6  2006    17
 7  2007    15
 8  2008     9
 9  2009     3
10  2010    16
11  2011    12
12  2012    21
13  2013    13
14  2014     1
15  2015    12
16  2016    22
```

There are various things you could say, most of which are likely to be good. My immediate reaction is that most of the years with a lot of heat-alert days are in the last few years, and

most of the years with not many are near the start, so there is something of an upward trend. Having said that, 2014 is unusually low (that was a cool summer), and 2005 was unusually high. (Was that the summer of the big power outage? I forget.[24]

You could also reasonably say that there isn't much pattern: the number of heat-alert days goes up and down. In fact, anything that's not obviously nonsense will do.

I was thinking about making a graph of these frequencies against year, and sticking some kind of smooth trend on it. This uses the output we just got, which is itself a data frame:

```
heat %>%
  count(year) %>%
  ggplot(aes(x = year, y = n)) + geom_point() + geom_smooth(se = F)
```

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

The pattern is very scattered, as is commonly the case with environmental-science-type data, but there is a very small upward trend. So it seems that either answer is justified, either "there is no trend" or "there is something of an upward trend".

[24] I looked it up. It was 2003, my first summer in Ontario. I realize as I write this that you may not be old enough to remember these years. Sometimes I forget how old I am.

The other thing I notice on this plot is that if there are a lot of heat-alert days one year, there will probably also be a lot in the next year (and correspondingly if the number of heat-alert days is below average: it tends to be below average again in the next year). This pattern is known to time-series people as "autocorrelation" and indicates that the number of heat-alert days in one year and the next is not independent: if you know one year, you can predict the next year. (Assessment of trend and autocorrelation are hard to untangle properly.)

Extra 1: I learn from Environmental Science grad students (of whom we have a number at UTSC) that the approved measure of association is called the Mann-Kendall correlation, which is the Kendall correlation of the data values with time. In the same way that we use the sign test when we doubt normality, and it uses the data more crudely but safely, the regular (so-called Pearson) correlation assumes normality (of the errors in the regression of one variable on the other), and when you doubt that (as you typically do with this kind of data) you compute a different kind of correlation with time. What the Kendall correlation does is to take each pair of observations and ask whether the trend with time is uphill or downhill. For example, there were 3 heat-alert days in 2009, 16 in 2010 and 12 in 2011. Between 2009 and 2010, the trend is uphill (increasing with time), and also between 2009 and 2011 (there were more heat-alert days in the later year), but between 2010 and 2011 the trend is downhill. The idea of the Kendall correlation is you take *all* the pairs of points, of which there are typically rather a lot, count up how many pairs are uphill and how many downhill, and apply a formula to get a correlation between -1 and 1. (If there are about an equal number of uphills and downhills, the correlation comes out near 0; if they are mostly uphill, the correlation is near 1, and if they are mostly downhill, the correlation is near -1.) It doesn't matter *how* uphill or downhill the trends are, only the number of each, in the same way that the sign test only counts the *number* of values above or below the hypothesized median, not how far above or below they are.

This can be calculated, and even tested:

```
heat %>%
  count(year) %>%
  with(., cor.test(year, n, method = "kendall"))
```

```
Warning in cor.test.default(year, n, method = "kendall"): Cannot compute exact
p-value with ties


    Kendall's rank correlation tau

data:  year and n
z = 0.31612, p-value = 0.7519
alternative hypothesis: true tau is not equal to 0
sample estimates:
```

```
       tau
0.05907646
```

The Mann-Kendall correlation is a thoroughly unremarkable 0.06, and with only 16 data points, a null hypothesis that the correlation is zero is far from being rejected, P-value 0.7519 as shown. So this is no evidence of a time trend at all.

Extra 2: I'd like to say a word about how I got these data. They came from link. If you take a look there, there are no obvious rows and columns. This format is called JSON. Look a bit more carefully and you'll see stuff like this, repeated:

```
{"id":"232","date":"2016-09-08","code":"HAU",
"text":"Toronto's Medical Officer of Health has upgraded the Heat Warning to an Extended Heat
```

one for each heat alert day. These are "keys" (on the left side of the `:`) and "values" (on the right side).[25] The keys are column headers (if the data were in a data frame) and the values are the data values that would be in that column. In JSON generally, there's no need for the keys to be the same in every row, but if they are, as they are here, the data can be arranged in a data frame. How? Read on.

I did this in R, using a package called `jsonlite`, with this code:

```
library(jsonlite)
url <- "http://app.toronto.ca/opendata/heat_alerts/heat_alerts_list.json"
heat <- fromJSON(url, simplifyDataFrame = T)
head(heat)
write_csv(heat, "heat.csv")
```

After loading the package, I create a variable `url` that contains the URL for the JSON file. The `fromJSON` line takes something that is JSON (which could be in text, a file or a URL) and converts it to and saves it in a data frame. Finally, I save the data frame in a `.csv` file. That's the `.csv` file you used. If you run that code, you'll get a `.csv` file of heat alerts right up to the present, and you can update my analysis.

Why `.csv`? If I had used `write_delim`, the values would have been separated by spaces. *But*, the `text` is a sentence of several words, which are themselves separated by spaces. I could have had you read in everything else and not the text, and then separated-by-spaces would have been fine, but I wanted you to see the text so that you could understand the `code` values. So `.csv` is what it was.

■

[25] This is the same kind of thing as a "dictionary" in Python.

15.23 Isoflavones

The plant called kudzu was imported to the US South from Japan. It is rich in isoflavones, which are believed to be beneficial for bones. In a study, rats were randomly assigned to one of three diets: one with a low dose of isoflavones from kudzu, one with a high dose, and a control diet with no extra isoflavones. At the end of the study, each rat's bone density was measured, in milligrams per square centimetre. The data as recorded are shown in http://ritsokiguess.site/isoflavones.txt.[26] There are 15 observations for each treatment, and hence 45 altogether.

Here are some code ideas you might need to use later, all part of the `tidyverse`. You may need to find out how they work.

- `col_names` (in the `read_` functions)
- `convert` (in various `tidyverse` functions)
- `fill`
- `na_if`
- `rename`
- `separate_rows`
- `skip` (in the `read_` functions)
- `values_drop_na` (in the `pivot_` functions)

If you use any of these, *cite* the webpage(s) or other source(s) where you learned about them.

(a) Take a look at the data file. Describe briefly what you see.

Solution

The data values are (at least kind of) aligned in columns, suggesting `read_table`. There are up to six bone density values in each row, with a header that spans all of them (by the looks of it). The treatment column looks all right except that some of the rows are blank. The blank treatments are the same as the ones in the row(s) above them, you can infer, because there are 15 observations for each treatment, six, six, and then three. (This is how a spreadsheet is often laid out: blank means the same as the previous line.)[27]

This, you might observe, will need some tidying.

■

[26] Evidently the units were chosen for ease of recording; had the values been in grams instead, the person recording the data would have had to put a 0 and a decimal point on the front of each value. This is the old meaning of the word coding; making the data values be whole numbers and/or small deviations from something makes them easier to record, and in pre-computer days easier to calculate with. You will also see the same word used for classifying survey responses into categories, which is similar but not quite the same thing.

[27] It shouldn't be, but it often is.

(b) Read in the data, using `read_table`, and get it into a tidy form, suitable for making a graph. This means finishing with (at least) a column of treatments with a suitable name (the treatments will be text) and a column of bone density values (numbers), one for each rat. You can have other columns as well; there is no obligation to get rid of them. Describe your process clearly enough that someone new to this data set would be able to understand what you have done and reproduce it on another similar dataset. Before you begin, think about whether or not you want to keep the column headers that are in the data file or not. (It can be done either way, but one way is easier than the other.)

Solution

The tidying part is a fair bit easier to see if you *do not* read the column headers. A clue to this is that `bone_mineral_density` is not aligned with the values (of bone mineral density) below it. The next question is how to do that. You might remember `col_names=FALSE` from when the data file has no column headers at all, but here it *does* have headers; we just want to skip over them. Keep reading in the documentation for `read_table`, and you'll find an option `skip` that does exactly that, leading to:

```
my_url <- "http://ritsokiguess.site/datafiles/isoflavones.txt"
bmd0a <- read_table(my_url, col_names = FALSE, skip = 1)
```

```
-- Column specification --------------------------------------------------------
cols(
  X1 = col_character(),
  X2 = col_double(),
  X3 = col_double(),
  X4 = col_double(),
  X5 = col_double(),
  X6 = col_double(),
  X7 = col_double()
)

Warning: 6 parsing failures.
row col  expected    actual                                                  file
  2  -- 7 columns 6 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  3  -- 7 columns 3 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  5  -- 7 columns 6 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  6  -- 7 columns 3 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  8  -- 7 columns 6 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
... ... ......... ......... ......................................................
See problems(...) for more details.
```

```r
bmd0a
```

```
# A tibble: 9 x 7
  X1           X2    X3    X4    X5    X6    X7
  <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 control     228   207   234   220   217   228
2 209         221   204   220   203   219    NA
3 218         245   210    NA    NA    NA    NA
4 low_dose    211   220   211   233   219   233
5 226         228   216   225   200   208    NA
6 198         208   203    NA    NA    NA    NA
7 high_dose   250   237   217   206   247   228
8 245         232   267   261   221   219    NA
9 232         209   255    NA    NA    NA    NA
```

If you miss the `skip`, the first row of "data" will be those column headers that were in the data file, and you really don't want that. This link talks about both `col_names` and `skip`.

This, however, is looking very promising. A pivot_longer will get those columns of numbers into one column, which we can call something like `bmd`, and …but, not so fast. What about those blank treatments in `X1`? The first two blank ones are `control`, the next two are `low_dose` and the last two are `high_dose`. How do we fill them in? The word "fill" might inspire you to read up on `fill`. Except that this doesn't quite work, because it replaces *missings* with the non-missing value above them, and we have blanks, not missings.

All right, can we replace the blanks with missings, and then `fill` those? This might inspire you to go back to the list of ideas in the question, and find out what `na_if` does: namely, exactly this! Hence:

```r
bmd0a %>% mutate(X1=na_if(X1, "")) %>%
fill(X1)
```

```
# A tibble: 9 x 7
  X1           X2    X3    X4    X5    X6    X7
  <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 control     228   207   234   220   217   228
2 209         221   204   220   203   219    NA
3 218         245   210    NA    NA    NA    NA
4 low_dose    211   220   211   233   219   233
5 226         228   216   225   200   208    NA
6 198         208   203    NA    NA    NA    NA
7 high_dose   250   237   217   206   247   228
```

```
8 245          232   267   261   221   219    NA
9 232          209   255    NA    NA    NA    NA
```

Run this one line at a time to see how it works. `fill` takes a column with missing values to replace, namely `X1`, and `na_if` takes two things: a column containing some values to make `NA`, and the values that should be made `NA`, namely the blank ones.

So that straightens out the treatment column. It needs renaming; you can do that now, or wait until later. I'm going to wait on that.

You need to organize the treatment column first, before you do the `pivot_longer`, or else that won't work.[28]

Now, we need to get one column of bone mass densities, instead of six. This you'll recognize as a standard `pivot_longer`, with one tweak: those missing values in `X5` through `X7`, which we want to get rid of. You might remember that this is what `values_drop_na` does:

```
bmd0a %>% mutate(X1=na_if(X1, "")) %>%
fill(X1) %>%
pivot_longer(X2:X7, names_to="old", values_to="bmd", values_drop_na=TRUE)
```

```
# A tibble: 39 x 3
   X1      old     bmd
   <chr>   <chr> <dbl>
 1 control X2      228
 2 control X3      207
 3 control X4      234
 4 control X5      220
 5 control X6      217
 6 control X7      228
 7 209     X2      221
 8 209     X3      204
 9 209     X4      220
10 209     X5      203
# i 29 more rows
```

If you didn't think of `values_drop_na`, do the pivot without, and then check that you have too many rows because the missings are still there (there are 45 rats but you have 54 rows), so add a `drop_na()` to the end of your pipe. The only missing values are in the column I called `bmd`.

[28] Data tidying has a lot of this kind of thing: try something, see that it doesn't work, figure out what went wrong, fix that, repeat. The work you hand in, or show to your boss, won't necessarily look very much like your actual process.

This is almost there. We have a numeric column of bone mass densities, a column called `old` that we can ignore, and a treatment column with a stupid name that we can fix. I find `rename` backwards: the syntax is new name equals old name, so you start with the name that doesn't exist yet and finish with the one you want to get rid of:

```
bmd0a %>% mutate(X1=na_if(X1, "")) %>%
fill(X1) %>%
pivot_longer(X2:X7, names_to="old", values_to="bmd", values_drop_na=TRUE) %>%
rename(treatment=X1) -> bmd1b
bmd1b
```

```
# A tibble: 39 x 3
   treatment old     bmd
   <chr>     <chr> <dbl>
 1 control   X2      228
 2 control   X3      207
 3 control   X4      234
 4 control   X5      220
 5 control   X6      217
 6 control   X7      228
 7 209       X2      221
 8 209       X3      204
 9 209       X4      220
10 209       X5      203
# i 29 more rows
```

Done!

The best way to describe this kind of work is to run your pipeline up to a point that needs explanation, describe what comes next, and then run the *whole pipeline* again up to the next point needing explanation, rinse and repeat. (This avoids creating unnecessary temporary dataframes, since the purpose of the pipe is to avoid those.)

The guideline for description is that if *you* don't know what's going to happen next, your reader won't know either. For me, that was these steps:

- read the data file without row names and see how it looks
- fix up the treatment column (convincing myself and the reader that we were now ready to pivot-longer)
- do the `pivot_longer` and make sure it worked
- rename the treatment column

So, I said there was another way. This happens to have a simple but clever solution. It starts from wondering "what happens if I read the data file *with* column headers, the normal way? Do it and find out:

```
my_url <- "http://ritsokiguess.site/datafiles/isoflavones.txt"
bmd0b <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  treatment = col_character(),
  bone_mineral_density = col_double()
)

Warning: 9 parsing failures.
row col  expected    actual                                                 file
  1  -- 2 columns 7 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  2  -- 2 columns 6 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  3  -- 2 columns 3 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  4  -- 2 columns 7 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
  5  -- 2 columns 6 columns 'http://ritsokiguess.site/datafiles/isoflavones.txt'
... ... ......... ......... ......................................................
See problems(...) for more details.
```

```
bmd0b
```

```
# A tibble: 9 x 2
  treatment bone_mineral_density
  <chr>                     <dbl>
1 control                     228
2 209                         221
3 218                         245
4 low_dose                    211
5 226                         228
6 198                         208
7 high_dose                   250
8 245                         232
9 232                         209
```

This looks … strange. There are two column headers, and so there are two columns. It so happened that this worked because the text `bone_mineral_density` is long enough to span

all the columns of numbers. That second column is actually *text*: six or three numbers as text with spaces between them.

The first thing is, as before, to fill in the missing treatments, which is as above, but changing some names:

```
bmd0b %>% mutate(treatment=na_if(treatment, "")) %>%
fill(treatment)
```

```
# A tibble: 9 x 2
  treatment bone_mineral_density
  <chr>                     <dbl>
1 control                     228
2 209                         221
3 218                         245
4 low_dose                    211
5 226                         228
6 198                         208
7 high_dose                   250
8 245                         232
9 232                         209
```

The way we learned in class for dealing with this kind of thing is `separate`. It is rather unwieldy here since we have to split `bone_mineral_density` into six (temporary) things:

```
bmd0b %>% mutate(treatment=na_if(treatment, "")) %>%
fill(treatment) %>%
separate(bone_mineral_density, into = c("z1", "z2", "z3", "z4", "z5", "z6"))
```

```
Warning: Expected 6 pieces. Missing pieces filled with `NA` in 9 rows [1, 2, 3, 4, 5, 6,
7, 8, 9].
```

```
# A tibble: 9 x 7
  treatment z1    z2    z3    z4    z5    z6
  <chr>     <chr> <chr> <chr> <chr> <chr> <chr>
1 control   228   <NA>  <NA>  <NA>  <NA>  <NA>
2 209       221   <NA>  <NA>  <NA>  <NA>  <NA>
3 218       245   <NA>  <NA>  <NA>  <NA>  <NA>
4 low_dose  211   <NA>  <NA>  <NA>  <NA>  <NA>
5 226       228   <NA>  <NA>  <NA>  <NA>  <NA>
6 198       208   <NA>  <NA>  <NA>  <NA>  <NA>
```

```
7 high_dose 250   <NA>  <NA>  <NA>  <NA>  <NA>
8 245       232   <NA>  <NA>  <NA>  <NA>  <NA>
9 232       209   <NA>  <NA>  <NA>  <NA>  <NA>
```

This works, though if you check, there's a warning that some of the rows don't have six values. However, these have been replaced by missings, which is just fine. From here, we do exactly what we did before: pivot-longer all the columns I called `z`-something, and get rid of the missings.

Having thought of `separate`, maybe you're now wondering what `separate_rows` does. It turns out that it bypasses the business of creating extra columns and then pivoting them longer, thus:

```
bmd0b %>% mutate(treatment=na_if(treatment, "")) %>%
fill(treatment)  %>%
separate_rows(bone_mineral_density, convert = TRUE) -> bmd1a
bmd1a
```

```
# A tibble: 9 x 2
  treatment bone_mineral_density
  <chr>                     <int>
1 control                     228
2 209                         221
3 218                         245
4 low_dose                    211
5 226                         228
6 198                         208
7 high_dose                   250
8 245                         232
9 232                         209
```

Boom! This takes all the things in that mess in `bone_mineral_density`, splits them up into individual data values, and puts them one per row back into the same column. The `convert` is needed because otherwise the values in the second column would be text and you wouldn't be able to plot them. (If you don't see that, use a mutate to convert the column into the numerical version of itself.)

■

(c) The statistician on this study is thinking about running an ordinary analysis of variance to compare the bone mineral density for the different treatments. Obtain a plot from your tidy dataframe that will help her decide whether that is a good idea.

Solution

The key issues here are whether the values within each treatment group are close enough to normally distributed, and, if they are, whether the spreads are close enough to equal. The best plot is therefore a normal quantile plot of each of the three groups, in facets. You can do this *without* `scales="free"`:

```
ggplot(bmd1b, aes(sample=bmd)) + stat_qq() + stat_qq_line() +
facet_wrap(~treatment)
```

The value of doing it this way is that you also get a sense of variability, from the slopes of the lines, or from how much of each box is filled vertically. (Here, the high-dose values are more spread-out than the other two groups, which are similar in spread.)

You could also do it *with* `scales = "free"`:

```
ggplot(bmd1b, aes(sample=bmd)) + stat_qq() + stat_qq_line() +
facet_wrap(~treatment, scales = "free")
```

The value of doing it *this* way is that you fill the facets (what I called "not wasting real estate" on an earlier assignment), and so you get a better assessment of normality, but the downside is that you will need another plot, for example a boxplot (see below) to assess equality of spreads if you are happy with the normality.

I'm happy with either way of making the normal quantile plots, as long as you have a *reason* for your choice, coming from what you will be using the normal quantile plot for. You might not think of saying that here as you do it, but when you do the next part, you may realize that you need to assess equality of spreads, and in that case you should come back here and add a reason for using or not using `scales = "free"`.

The next-best graph here is boxplots:

```
ggplot(bmd1b, aes(x=treatment, y=bmd)) + geom_boxplot()
```

This is not so good because it doesn't address normality as directly (just giving you a general sense of shape). On the other hand, you can assess spread directly with a boxplot; see discussion above.

The grader is now probably thoroughly confused, so let me summarize possible answers in order of quality:

1. A normal quantile plot of all three groups, using `scales = "free"` or not, with a good reason. (If with `scales = "free"`, and there needs to be a comparison of spread, there needs to be a boxplot or similar below as well. That's what I meant by "any additional graphs" in the next part.)
2. A normal quantile plot of all three groups, using `scales = "free"` or not, *without* a good reason.
3. A side-by-side boxplot. Saying in addition that normality doesn't matter so much because we have moderate-sized samples of 15 and therefore that boxplots are good enough moves this answer up a place.

Note that getting the graph is (relatively) easy once you have the tidy data, but is impossible if you don't! This is the way the world of applied statistics works; without being able to get your data into the right form, you won't be able to do *anything* else. This question is consistent with that fact; I'm not going to give you a tidy version of the data so that you can make some graphs. The point of this question is to see whether you can get the data tidy enough, and if you can, you get the bonus of being able to do something straightforward with it.

■

(d) Based on your graph, and any additional graphs you wish to draw, what analysis would you recommend for this dataset? Explain briefly. (Don't do the analysis.)

Solution

Make a decision about normality first. You need *all three* groups to be sufficiently normal. I don't think there's any doubt about the high-dose and low-dose groups; these are if anything *short*-tailed, which is not a problem for the ANOVA. You might find that the control group is OK too; make a call. Or you might find it skewed to the right, something suggested rather more by the boxplot. My take, from looking at the normal quantile plot, is that the highest value in the control group is a little too high, but with a sample size of 15, the Central Limit Theorem will take care of that. For yourself, you can find a bootstrapped sampling distribution of the sample mean for the control group and see how normal it looks.

If you are not happy with the normality, recommend Mood's median test.

If you are OK with the normality, you need to assess equal spreads. You can do this from a boxplot, where the high-dose group clearly has bigger spread. Or, if you drew normal quantile plots *without* `scales = "free"`, compare the slopes of the lines. This means that you need to recommend a Welch ANOVA.

If your normal quantile plots looked like this:

```
ggplot(bmd1b, aes(sample=bmd)) + stat_qq() + stat_qq_line() +
facet_wrap(~treatment, scales = "free")
```

the only way to assess spread is to make another plot, and for this job, the boxplot is best.

Extra 1: the bootstrapped sampling distribution of the sample mean for the control group goes this way:

```
bmd1b %>%
filter(treatment == "control") -> d
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(d$bmd, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```

No problems there. The Welch ANOVA is fine.

Extra 2: You might be curious how the analysis comes out. Here is Welch:

```
oneway.test(bmd~treatment, data=bmd1b)
```

```
    One-way analysis of means (not assuming equal variances)

data:  bmd and treatment
```

```
F = 2.5385, num df = 8.000, denom df = 7.718, p-value = 0.1082
```

Not all the same means, so use Games-Howell to explore:

```r
gamesHowellTest(bmd~factor(treatment), data = bmd1b)
```

```
Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced
```

```
Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced

Warning in ptukey(abs(qval), nmeans = k, df = df, lower.tail = FALSE): NaNs
produced


    Pairwise comparisons using Games-Howell test

data: bmd by factor(treatment)

          198  209  218 226  232 245  control high_dose
209       0.75 -    -   -    -   -    -       -
218       -    -    -   -    -   -    -       -
226       0.73 1.00 -   -    -   -    -       -
232       -    -    -   -    -   -    -       -
245       0.20 0.41 -   0.51 -   -    -       -
control   0.13 0.80 -   0.97 -   0.77 -       -
high_dose 0.15 0.51 -   0.70 -   1.00 0.97    -
low_dose  0.18 0.89 -   0.99 -   0.72 1.00    0.94


P value adjustment method: none

alternative hypothesis: two.sided
```

High dose is significantly different from both the other two, which are not significantly different from each other.

Mood's median test, for comparison:

```
median_test(bmd1b, bmd, treatment)
```

```
$table
           above
group       above below
  198           0     2
```

```
  209           1     3
  218           1     1
  226           2     3
  232           1     1
  245           4     1
  control       3     2
  high_dose     4     2
  low_dose      2     3

$test
       what     value
1 statistic 6.0666667
2        df 8.0000000
3   P-value 0.6397643
```

Not *any* significant differences, although it is a close thing.

The table of aboves and belows suggests the same thing as the Welch test: the high-dose values are mainly high, and the others are mostly low. But with these sample sizes it is not strong enough evidence. My guess is that the median test is lacking power compared to the Welch test; having seen that the Welch test is actually fine, it is better to use that here.[29]

■

15.24 Jocko's Garage

Insurance adjusters are concerned that Jocko's Garage is giving estimates for repairing car damage that are too high. To see whether this is indeed the case, ten cars that had been in collisions were taken to both Jocko's Garage and another garage, and the two estimates for repair were recorded. The data as recorded are here.

(a) Take a look at the data file (eg. by using your web browser). How are the data laid out? Do there appear to be column headers?

Solution

The data are laid out in aligned columns, so that we will need to use `read_table` to read it in. There are no column headers, since there is no line at the top of the file saying what each column represents. (The fact that I was asking about column headers is kind of a clue that something non-standard is happening there.)

[29]This is the opposite way to the usual: when two tests disagree, it is usually the one with fewer assumptions that is preferred, but in this case, the Welch ANOVA is fine, and the median test fails to give significance because it is not using the data as efficiently.

■

(b) Read in and display the data file, bearing in mind what you just concluded about it. What names did the columns acquire?

Solution

As mentioned above, you'll need `read_table`, plus `col_names=FALSE` to *not* read the first row as column names:

```
my_url <- "http://ritsokiguess.site/datafiles/jocko.txt"
cars0 <- read_table(my_url, col_names = FALSE)
```

```
-- Column specification --------------------------------------------------------
cols(
  X1 = col_character(),
  X2 = col_character(),
  X3 = col_double(),
  X4 = col_double(),
  X5 = col_double(),
  X6 = col_double(),
  X7 = col_double()
)
```

```
cars0
```

```
# A tibble: 6 x 7
  X1    X2       X3    X4    X5    X6    X7
  <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
1 a     Car       1     2     3     4     5
2 a     Jocko  1375  1550  1250  1300   900
3 a     Other  1250  1300  1250  1200   950
4 b     Car       6     7     8     9    10
5 b     Jocko  1500  1750  3600  2250  2800
6 b     Other  1575  1600  3300  2125  2600
```

The column names have become `X1` through `X7`. You'll need to work with these in a minute, so it is good to be aware of that now.

I used a "temporary" name for my dataframe since we are going to be tidying it before we do anything with it, and I'm saving the "good" name `cars` for the tidy one.

■

(c) Make this data set tidy. That is, you need to end up with columns containing the repair cost estimates at each of the two garages and also identifying the cars, with each observation on one row. Describe your thought process. (It needs to be possible for the reader to follow your description and understand why it works.) Save your tidy dataframe.

Solution

This looks very far from tidy right now. The things in `X2` look like they will need to be variable names eventually, but there are two copies of them, and there are also five columns of data values that need eventually to become three. Having all the data values in *one* column might be a useful place to start:

```
cars0 %>% pivot_longer(X3:X7, names_to="old_cols", values_to="values")
```

```
# A tibble: 30 x 4
   X1    X2    old_cols values
   <chr> <chr> <chr>     <dbl>
 1 a     Car   X3            1
 2 a     Car   X4            2
 3 a     Car   X5            3
 4 a     Car   X6            4
 5 a     Car   X7            5
 6 a     Jocko X3         1375
 7 a     Jocko X4         1550
 8 a     Jocko X5         1250
 9 a     Jocko X6         1300
10 a     Jocko X7          900
# i 20 more rows
```

This is tidier, but it's now *too long*: this has 30 rows but there are only 10 cars, or, depending on your point of view, there are 20 observations on 10 individual cars, so you could justify (in some way) having 20 rows, but not 30.

Now, therefore, we need to pivot *wider*. But to get to this point, we had to try pivoting longer to see what it did, and then go from there. I don't think it's at all obvious that this is what will happen, so I think you need to do a pivot-longer first, *talk about it*, and then move on.

From here, we want to make columns whose names are the things in `X2`, and whose values are the things in `values`. This is exactly what `pivot_wider` does, so add that to our pipe:

```
cars0 %>% pivot_longer(X3:X7, names_to="names", values_to="values") %>%
pivot_wider(names_from = X2, values_from = values) -> cars
```

```r
cars
```

```
# A tibble: 10 x 5
   X1    names   Car Jocko Other
   <chr> <chr> <dbl> <dbl> <dbl>
 1 a     X3        1  1375  1250
 2 a     X4        2  1550  1300
 3 a     X5        3  1250  1250
 4 a     X6        4  1300  1200
 5 a     X7        5   900   950
 6 b     X3        6  1500  1575
 7 b     X4        7  1750  1600
 8 b     X5        8  3600  3300
 9 b     X6        9  2250  2125
10 b     X7       10  2800  2600
```

This is now tidy: one row for each of the 10 cars, one column containing the repair estimates for each car at each of the two garages, and a column identifying the cars. I think this is best because this is a matched-pairs study, and so you want the two measurements for each individual car in columns next to each other (for `t.test` with `paired=TRUE`).

I think it is best to show the whole pipeline here, even though you are making R work a little harder, rather than having to make up a temporary variable name for the output from `pivot_longer` (that you are never going to look at again after this).

If you thought there were 20 observations, you have a bit more work to do (that you will have to undo later to get the right graph), namely:

```r
cars %>% pivot_longer(c(Jocko, Other), names_to="garage", values_to="estimate") -> cars1
cars1
```

```
# A tibble: 20 x 5
   X1    names   Car garage estimate
   <chr> <chr> <dbl> <chr>     <dbl>
 1 a     X3        1 Jocko      1375
 2 a     X3        1 Other      1250
 3 a     X4        2 Jocko      1550
 4 a     X4        2 Other      1300
 5 a     X5        3 Jocko      1250
 6 a     X5        3 Other      1250
 7 a     X6        4 Jocko      1300
 8 a     X6        4 Other      1200
```

```
 9 a     X7        5 Jocko       900
10 a     X7        5 Other       950
11 b     X3        6 Jocko      1500
12 b     X3        6 Other      1575
13 b     X4        7 Jocko      1750
14 b     X4        7 Other      1600
15 b     X5        8 Jocko      3600
16 b     X5        8 Other      3300
17 b     X6        9 Jocko      2250
18 b     X6        9 Other      2125
19 b     X7       10 Jocko      2800
20 b     X7       10 Other      2600
```

This would be the right thing to do if you had independent observations (that is, *20 different* cars, and you randomly choose a garage to send each one to). But you can have a car assessed for repair without actually repairing it, so it makes more sense to send each car to both garages, and compare like with like. Compare the kids learning to read; once a child has learned to read, you can't teach them to read again, so that study had to be done with two independent samples.

Extra: I thought about starting by making the dataframe even wider:

```r
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7)
```

```
# A tibble: 2 x 16
  X1    X3_Car X3_Jocko X3_Other X4_Car X4_Jocko X4_Other X5_Car X5_Jocko
  <chr>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>
1 a          1     1375     1250      2     1550     1300      3     1250
2 b          6     1500     1575      7     1750     1600      8     3600
# i 7 more variables: X5_Other <dbl>, X6_Car <dbl>, X6_Jocko <dbl>,
#   X6_Other <dbl>, X7_Car <dbl>, X7_Jocko <dbl>, X7_Other <dbl>
```

This is sort of the right thing, but there are repeat columns now, depending on where the data values came from in `cars0`. What we want to do now is *some* kind of `pivot_longer`, creating three columns called `Car`, `Jocko`, and `Other`. If we only had *one* kind of thing to make longer, this would be a standard `pivot_longer`. But we have three. There are two "extras" to `pivot_longer` that will get you to the right place. The first one is to give multiple inputs to `names_to`, because the column names encode two things: where in the original data frame the value came from (which is now junk to us), and what the value actually represents, which we definitely *do* want to keep. I don't have a good name for it, though, so I'll call it `z` for now. Note that we need a `names_sep` that says what the two things in the column names are separated by, the underscore that the `pivot_wider` put in there:

```r
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7) %>%
pivot_longer(-X1, names_to = c("junk", "z"), names_sep="_")
```

```
# A tibble: 30 x 4
   X1    junk  z     value
   <chr> <chr> <chr> <dbl>
 1 a     X3    Car       1
 2 a     X3    Jocko  1375
 3 a     X3    Other  1250
 4 a     X4    Car       2
 5 a     X4    Jocko  1550
 6 a     X4    Other  1300
 7 a     X5    Car       3
 8 a     X5    Jocko  1250
 9 a     X5    Other  1250
10 a     X6    Car       4
# i 20 more rows
```

This is now exactly what I got by *starting* with `pivot_longer`, and so the same `pivot_wider` that I finished with before will tidy this up:

```r
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7) %>%
pivot_longer(-X1, names_to = c("junk", "z"), names_sep="_") %>%
pivot_wider(names_from = z, values_from = value)
```

```
# A tibble: 10 x 5
   X1    junk    Car Jocko Other
   <chr> <chr> <dbl> <dbl> <dbl>
 1 a     X3        1  1375  1250
 2 a     X4        2  1550  1300
 3 a     X5        3  1250  1250
 4 a     X6        4  1300  1200
 5 a     X7        5   900   950
 6 b     X3        6  1500  1575
 7 b     X4        7  1750  1600
 8 b     X5        8  3600  3300
 9 b     X6        9  2250  2125
10 b     X7       10  2800  2600
```

This is now tidy, so you have achieved what you set out to do, but you have not done it the best way, so you should expect to lose a little something.

This kind of longer-then-wider happens often enough that there is an option in `pivot_longer` to do it in one step. Let's remind ourselves of where we were:

```
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7)
```

```
# A tibble: 2 x 16
  X1    X3_Car X3_Jocko X3_Other X4_Car X4_Jocko X4_Other X5_Car X5_Jocko
  <chr>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>
1 a          1     1375     1250      2     1550     1300      3     1250
2 b          6     1500     1575      7     1750     1600      8     3600
# i 7 more variables: X5_Other <dbl>, X6_Car <dbl>, X6_Jocko <dbl>,
#   X6_Other <dbl>, X7_Car <dbl>, X7_Jocko <dbl>, X7_Other <dbl>
```

The second part of those funky column names needs to become *the names of our new columns*. To make that happen in one step, you put the special indicator `.value` in where we had `z` before:

```
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7) %>%
pivot_longer(-X1, names_to = c("junk", ".value"), names_sep="_")
```

```
# A tibble: 10 x 5
   X1    junk    Car Jocko Other
   <chr> <chr> <dbl> <dbl> <dbl>
 1 a     X3        1  1375  1250
 2 a     X4        2  1550  1300
 3 a     X5        3  1250  1250
 4 a     X6        4  1300  1200
 5 a     X7        5   900   950
 6 b     X3        6  1500  1575
 7 b     X4        7  1750  1600
 8 b     X5        8  3600  3300
 9 b     X6        9  2250  2125
10 b     X7       10  2800  2600
```

and as if by magic, we have tidiness. It's best to discover this and do it in two steps, though by starting with `pivot_wider` you have made it more difficult for yourself. By starting with `pivot_longer`, it is a very standard longer-then-wider, and there is nothing extra you have to learn. (The column `X1` I added to the data so that `pivot_wider` would work smoothly. See what happens if you remove it with `select(-X1)` before you start pivoting.)

There is usually a relatively simple way to do these, and if your way is complicated, that is an invitation to try it again a different way. I don't think there's a way to do it in *one* step, though, because those things in `X2` have to get to column names somehow, and they can only do so by being attached to which original column the values came from.

All of these ideas are here, which is a dense read, but worth working through to see what is possible. This problem is of the type in "Longer, then wider".

■

(d) Make a suitable graph to assess the comparison of interest, and say briefly what your graph tells you.

Solution

You might be tempted to look at `cars`, see two quantitative variables, and think "scatterplot":

```
ggplot(cars, aes(x=Jocko, y=Other)) + geom_point()
```

This says that a repair that is more expensive at one garage is more expensive at the other as well, which is true, but it's an answer to the *wrong question*. We care about whether Jocko's Garage is more expensive than the other one *on the same car*. To rescue the scatterplot, you can add the line $y = x$ to the graph and see which side of the line the points are, which you might have to find out how to do:

```
ggplot(cars, aes(x=Jocko, y=Other)) + geom_point() + geom_abline(slope = 1, intercept = 0)
```

More of the points are below and to the right of the line, indicating that Jocko's Garage is typically more expensive (in the cases where the other garage is more expensive, there is not much in it).

There is a more direct approach here, based on the idea that a matched pairs test looks at the differences between the two estimates for each car: work out the differences, and make a *one*-sample plot of them:

```
cars %>% mutate(diff=Jocko-Other) %>%
ggplot(aes(x=diff)) + geom_histogram(bins = 4)
```

Most of the differences, this way around, are positive, so the indication is that Jocko's Garage is indeed more expensive. Don't have too many bins.

A one-sample boxplot of the differences would also work:

```
cars %>% mutate(diff=Jocko-Other) %>%
ggplot(aes(x=1, y=diff)) + geom_boxplot()
```

This tells you that at least 75% of the differences are positive.

If you ended up with my `cars1`:

```
cars1
```

```
# A tibble: 20 x 5
   X1    names   Car garage estimate
   <chr> <chr> <dbl> <chr>     <dbl>
 1 a     X3        1 Jocko      1375
 2 a     X3        1 Other      1250
 3 a     X4        2 Jocko      1550
 4 a     X4        2 Other      1300
 5 a     X5        3 Jocko      1250
 6 a     X5        3 Other      1250
 7 a     X6        4 Jocko      1300
 8 a     X6        4 Other      1200
 9 a     X7        5 Jocko       900
10 a     X7        5 Other       950
11 b     X3        6 Jocko      1500
12 b     X3        6 Other      1575
13 b     X4        7 Jocko      1750
14 b     X4        7 Other      1600
```

```
15 b     X5        8 Jocko      3600
16 b     X5        8 Other      3300
17 b     X6        9 Jocko      2250
18 b     X6        9 Other      2125
19 b     X7       10 Jocko      2800
20 b     X7       10 Other      2600
```

this is "obviously" a boxplot:

```
ggplot(cars1, aes(x=garage, y=estimate)) + geom_boxplot()
```

except that you have not used the fact that each group is measurements on the *same* 10 cars. Here is a way to rescue that:

```
ggplot(cars1, aes(x=garage, y=estimate, group=Car)) + geom_point() + geom_line()
```

The majority of the lines go downhill, so Jocko's Garage is more expensive most of the time. (The lines are really another way to look at the differences.) This last graph I would be good with, since it shows which pairs of measurements are related because of being on the same cars.

■

(e) Carry out a test to make an appropriate comparison of the mean estimates. What do you conclude, in the context of the data?

Solution

Comparing means requires the right flavour of t-test, in this case a matched-pairs one, with a one-sided alternative, since we were concerned that the Jocko estimates were bigger. In a matched pairs test, `alternative` says how the first column you name compares with the other one. If your columns are the opposite way to mine, your `alternative` needs to be "less":

```
with(cars, t.test(Jocko, Other, paired = TRUE, alternative = "greater"))
```

```
    Paired t-test

data:  Jocko and Other
t = 2.8749, df = 9, p-value = 0.009164
```

```
alternative hypothesis: true mean difference is greater than 0
95 percent confidence interval:
 40.76811      Inf
sample estimates:
mean difference
          112.5
```

Remember that this flavour of t-test doesn't take a `data=`, so you need to use `with` or dollar signs.

The P-value is actually just less than 0.01, so we can definitely conclude that the Jocko estimates are bigger on average.

If you calculated the differences earlier, feel free to use them here:

```
cars %>% mutate(diff=Jocko-Other) %>%
with(., t.test(diff, mu=0, alternative = "greater"))
```

```
    One Sample t-test

data:  diff
t = 2.8749, df = 9, p-value = 0.009164
alternative hypothesis: true mean is greater than 0
95 percent confidence interval:
 40.76811      Inf
sample estimates:
mean of x
    112.5
```

Saving the data frame with the differences in it is probably smart.

Again, if you got to `cars1`, you might think to do this:

```
t.test(estimate~garage, data=cars1, alternative="greater")
```

```
    Welch Two Sample t-test

data:  estimate by garage
t = 0.32056, df = 17.798, p-value = 0.3761
alternative hypothesis: true difference in means between group Jocko and group Other is grea
```

```
95 percent confidence interval:
 -496.4343       Inf
sample estimates:
mean in group Jocko mean in group Other
             1827.5              1715.0
```

but you would be *wrong*, because the two groups are not independent (they're the same cars at each garage). You have also lost the significant result, because some of the repairs are more expensive than others (at both garages), and this introduces extra variability that this test does not account for.

I said to compare the means, so I don't want a sign test here. If you think we should be doing one, you'll need to make the case for it properly: first, calculate and plot the differences and make the case that they're not normal enough. I see left-skewness in the histogram of differences, but personally I don't find this bad enough to worry about. If you do, make that case (but, a sample of size 10 even from a normal distribution might look this skewed) and then run the right test:

```r
cars %>% mutate(diff=Jocko-Other) %>%
sign_test(diff, 0)
```

```
$above_below
below above
    2     7

$p_values
  alternative    p_value
1       lower 0.98046875
2       upper 0.08984375
3   two-sided 0.17968750
```

The upper-tail P-value is the one you want (explain why), and this is not quite significant. This is different from the correct t-test for a couple of reasons: there is probably not much power with this small sample, and the two estimates that are higher at the Other garage are not much higher, which the t-test accounts for but the sign test does not.

■

15.25 Tidying electricity consumption

How does the consumption of electricity depend on temperature? To find out, a short-term study was carried out by a utility company based in a certain area. For a period of two

years, the average monthly temperature was recorded (in degrees Fahrenheit), the mean daily demand for electricity per household (in kilowatt hours), and the cost per kilowatt hour of electricity for that year (8 cents for the first year and 10 cents for the second, which it will be easiest to treat as categorical).

The data were laid out in an odd way, as shown in http://ritsokiguess.site/datafiles/utils.txt, in aligned columns: the twelve months of temperature were laid out on *two* lines for the first year, then the twelve months of consumption for the first year on the next two lines, and then four more lines for the second year laid out the same way. Thus the temperature of 31 in the first line goes with the consumption of 55 in the *third* line, and the last measurements for that year are the 78 at the end of the second line (temperature) and 73 at the end of the fourth line (consumption). Lines 5 through 8 of the data file are the same thing for the second year (when electricity was more expensive).

The data seem to have been laid out in order of temperature, rather than in order of months, which I would have thought would make more sense. But this is what we have.

(a) Read in and display the data file, bearing in mind that it has *no column names*.

Solution

That means `col_names = FALSE` when reading in. I gave this a "disposable" name, saving the good name `utils` for the tidy version:

```
my_url <- "http://ritsokiguess.site/datafiles/utils.txt"
utils0 <- read_table(my_url, col_names = FALSE)
```

```
-- Column specification --------------------------------------------------------
cols(
  X1 = col_character(),
  X2 = col_character(),
  X3 = col_double(),
  X4 = col_double(),
  X5 = col_double(),
  X6 = col_double(),
  X7 = col_double(),
  X8 = col_double()
)
```

```
utils0
```

```
# A tibble: 8 x 8
  X1      X2             X3    X4    X5    X6    X7    X8
  <chr>   <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 8cents  temperature    31    34    39    42    47    56
2 8cents  temperature    62    66    68    71    75    78
3 8cents  consumption    55    49    46    47    40    43
4 8cents  consumption    41    46    44    51    62    73
5 10cents temperature    32    36    39    42    48    56
6 10cents temperature    62    66    68    72    75    79
7 10cents consumption    50    44    42    42    36    40
8 10cents consumption    39    44    40    44    50    55
```

The columns have acquired names `X1` through `X8`. It doesn't really matter what these names *are*, but as we will see shortly, it matters that they *have* names.

■

(b) Arrange these data tidily, so that there is a column of price (per kilowatt hour), a column of temperatures, and a column of consumptions. Describe your process, including why you got list-columns (if you did) and what you did about them (if necessary).

Solution

This question is asking about your process as well as your answer, so I think it's best to build a pipeline one step at a time (which corresponds in any case to how you would figure out what to do). The first step seems to be to make longer, for example getting all those numbers in one column. I'm not quite sure what to call the new columns, so I'll make up some names and figure things out later:

```r
utils0 %>% pivot_longer(X3:X8, names_to = "col", values_to = "value")
```

```
# A tibble: 48 x 4
   X1     X2          col   value
   <chr>  <chr>       <chr> <dbl>
 1 8cents temperature X3       31
 2 8cents temperature X4       34
 3 8cents temperature X5       39
 4 8cents temperature X6       42
 5 8cents temperature X7       47
 6 8cents temperature X8       56
 7 8cents temperature X3       62
 8 8cents temperature X4       66
 9 8cents temperature X5       68
```

```
10 8cents temperature X6       71
# i 38 more rows
```

If you scroll down, `X2` has consumptions as well as temperatures, so we need to get that straightened out.

This, so far, is actually a lot like the weather one in lecture (where we had a max and a min temperature), and the solution is the same: follow up with a `pivot_wider` to get the temperatures and consumptions in their own columns:

```
utils0 %>% pivot_longer(X3:X8, names_to = "col", values_to = "value") %>%
pivot_wider(names_from = X2, values_from = value)
```

```
Warning: Values from `value` are not uniquely identified; output will contain list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} %>%
  dplyr::group_by(X1, col, X2) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
# A tibble: 12 x 4
   X1      col   temperature consumption
   <chr>   <chr> <list>      <list>
 1 8cents  X3    <dbl [2]>   <dbl [2]>
 2 8cents  X4    <dbl [2]>   <dbl [2]>
 3 8cents  X5    <dbl [2]>   <dbl [2]>
 4 8cents  X6    <dbl [2]>   <dbl [2]>
 5 8cents  X7    <dbl [2]>   <dbl [2]>
 6 8cents  X8    <dbl [2]>   <dbl [2]>
 7 10cents X3    <dbl [2]>   <dbl [2]>
 8 10cents X4    <dbl [2]>   <dbl [2]>
 9 10cents X5    <dbl [2]>   <dbl [2]>
10 10cents X6    <dbl [2]>   <dbl [2]>
11 10cents X7    <dbl [2]>   <dbl [2]>
12 10cents X8    <dbl [2]>   <dbl [2]>
```

Except that it didn't appear to work. Although it actually did. These are list-columns. I actually recorded lecture 14a to help you with this. (See also the discussion in Extra 3 of the last part of the writers question, and the word "list" at the top of `temperature` and `consumption`). Each cell holds *two* numbers instead of the one you were probably expecting.

Why did that happen? The warning above the output is a clue. Something is going to be "not uniquely identified". Think about how `pivot_wider` works. It has to decide which *row* and *column* of the wider dataframe to put each value in. The column comes from the `names_from`: temperature or consumption. So that's not a problem. The row comes from the combination of the other columns not named in the `pivot_wider`: that means the ones called `X1` and `col`. (Another way to see that is the columns in the result from the `pivot_wider` that *do not* have values in them: not `temperature` or `consumption`, the other two.)

If you look back at the things in `col`, they go from `X3` to `X8`, so there are six of them. There are two values in `X1`, so there are $2 \times 6 = 12$ combinations of the two, and so 12 rows in the wider dataframe. This has two columns, and thus $12 \times 2 = 24$ cells altogether. But there were 48 values in the longer dataframe (go back and look: it has 48 rows), so there isn't enough room for all of them here.

If you go back and look at the longer dataframe, you'll see, for example, that there are two `temperature` values that go with an `X1` of 8 cents and a col of `X3`, so that they will both have to be jammed into one cell of the wider dataframe.

The resolution of the list-columns here is the same as in the one about the writers: `unnest` them, and then you can ignore the warning:

```
utils0 %>% pivot_longer(X3:X8, names_to = "col", values_to = "value") %>%
pivot_wider(names_from = X2, values_from = value) %>%
unnest(c(temperature, consumption)) -> utils
```

```
Warning: Values from `value` are not uniquely identified; output will contain list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} %>%
  dplyr::group_by(X1, col, X2) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
utils
```

```
# A tibble: 24 x 4
   X1     col   temperature consumption
   <chr>  <chr>       <dbl>       <dbl>
 1 8cents X3             31          55
 2 8cents X3             62          41
 3 8cents X4             34          49
```

```
 4 8cents X4             66          46
 5 8cents X5             39          46
 6 8cents X5             68          44
 7 8cents X6             42          47
 8 8cents X6             71          51
 9 8cents X7             47          40
10 8cents X7             75          62
# i 14 more rows
```

There were 24 months of data, and a temperature and consumption for each, so this is now tidy and I can give it a proper name.

Extra: if you got to here and got scared:

```r
utils0 %>% pivot_longer(X3:X8, names_to = "col", values_to = "value") %>%
pivot_wider(names_from = X2, values_from = value)
```

```
Warning: Values from `value` are not uniquely identified; output will contain list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} %>%
  dplyr::group_by(X1, col, X2) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
# A tibble: 12 x 4
   X1      col   temperature consumption
   <chr>   <chr> <list>      <list>
 1 8cents  X3    <dbl [2]>   <dbl [2]>
 2 8cents  X4    <dbl [2]>   <dbl [2]>
 3 8cents  X5    <dbl [2]>   <dbl [2]>
 4 8cents  X6    <dbl [2]>   <dbl [2]>
 5 8cents  X7    <dbl [2]>   <dbl [2]>
 6 8cents  X8    <dbl [2]>   <dbl [2]>
 7 10cents X3    <dbl [2]>   <dbl [2]>
 8 10cents X4    <dbl [2]>   <dbl [2]>
 9 10cents X5    <dbl [2]>   <dbl [2]>
10 10cents X6    <dbl [2]>   <dbl [2]>
11 10cents X7    <dbl [2]>   <dbl [2]>
12 10cents X8    <dbl [2]>   <dbl [2]>
```

which is an entirely reasonable reaction, you might have asked yourself how you could have prevented this from happening. The problem, as discussed earlier, is with the rows, and that the `X1-col` combinations repeat. Let's go back to "longer":

```
utils0 %>% pivot_longer(X3:X8, names_to = "col", values_to = "value")
```

```
# A tibble: 48 x 4
   X1     X2          col   value
   <chr>  <chr>       <chr> <dbl>
 1 8cents temperature X3       31
 2 8cents temperature X4       34
 3 8cents temperature X5       39
 4 8cents temperature X6       42
 5 8cents temperature X7       47
 6 8cents temperature X8       56
 7 8cents temperature X3       62
 8 8cents temperature X4       66
 9 8cents temperature X5       68
10 8cents temperature X6       71
# i 38 more rows
```

Rows 1 and 7, 2 and 8, etc, are "replicates" in that they have the same `X1` and `col` values but different temperatures. This is because they come from the same column in the original layout of the data (the 31 and the 62 are underneath each other). This means that the first six rows are "replicate 1" and the next six are "replicate 2". Scrolling down, we then get to 8 cents and consumption, and we need to do the same again. So if we make a column that has 1s and 2s in the right places (six 1s, six 2s, repeat), we should then have unique rows for the `pivot_wider`.

```
utils0 %>% pivot_longer(X3:X8, names_to = "col", values_to = "value") %>%
mutate(replicate = rep(1:2, each = 6, length.out = 48))
```

```
# A tibble: 48 x 5
   X1     X2          col   value replicate
   <chr>  <chr>       <chr> <dbl>     <int>
 1 8cents temperature X3       31         1
 2 8cents temperature X4       34         1
 3 8cents temperature X5       39         1
 4 8cents temperature X6       42         1
 5 8cents temperature X7       47         1
 6 8cents temperature X8       56         1
```

```
 7 8cents temperature X3       62         2
 8 8cents temperature X4       66         2
 9 8cents temperature X5       68         2
10 8cents temperature X6       71         2
# i 38 more rows
```

`rep` does repeats like this: something to repeat (the numbers 1 through 2), how many times to repeat each one (six times), and how long the final thing has to be (48, since there were 48 rows in the longer dataframe).

Then, this time, if we do the `pivot_wider`, it should give us something tidy:

```
utils0 %>% pivot_longer(X3:X8, names_to = "col", values_to = "value") %>%
mutate(replicate = rep(1:2, each = 6, length.out = 48)) %>%
pivot_wider(names_from = X2, values_from = value)
```

```
# A tibble: 24 x 5
   X1     col   replicate temperature consumption
   <chr>  <chr>     <int>       <dbl>       <dbl>
 1 8cents X3            1          31          55
 2 8cents X4            1          34          49
 3 8cents X5            1          39          46
 4 8cents X6            1          42          47
 5 8cents X7            1          47          40
 6 8cents X8            1          56          43
 7 8cents X3            2          62          41
 8 8cents X4            2          66          46
 9 8cents X5            2          68          44
10 8cents X6            2          71          51
# i 14 more rows
```

and so it does, with 24 rows for the 24 months.

Another, perhaps easier, way to think about this (you might find it easier, anyway) is to go back to the original dataframe and make the `replicate` there:

```
utils0
```

```
# A tibble: 8 x 8
  X1      X2             X3    X4    X5    X6    X7    X8
  <chr>   <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 8cents  temperature    31    34    39    42    47    56
```

```
2 8cents  temperature    62    66    68    71    75    78
3 8cents  consumption    55    49    46    47    40    43
4 8cents  consumption    41    46    44    51    62    73
5 10cents temperature    32    36    39    42    48    56
6 10cents temperature    62    66    68    72    75    79
7 10cents consumption    50    44    42    42    36    40
8 10cents consumption    39    44    40    44    50    55
```

The first two rows are replicates (both 8 cents and temperature), then the third and fourth, and so on. So setting a `replicate` column as 1, 2, 1, 2 etc should do it, and this is short enough to type directly. Do this *first*, then the `pivot_longer`, then the `pivot_wider` as we did before, and we should end up with something tidy:

```
utils0 %>% mutate(replicate = c(1,2,1,2,1,2,1,2)) %>%
pivot_longer(X3:X8, names_to = "col", values_to = "value") %>%
pivot_wider(names_from = X2, values_from = value) %>%
unnest(c(temperature, consumption))
```

```
# A tibble: 24 x 5
   X1     replicate col   temperature consumption
   <chr>      <dbl> <chr>       <dbl>       <dbl>
 1 8cents         1 X3             31          55
 2 8cents         1 X4             34          49
 3 8cents         1 X5             39          46
 4 8cents         1 X6             42          47
 5 8cents         1 X7             47          40
 6 8cents         1 X8             56          43
 7 8cents         2 X3             62          41
 8 8cents         2 X4             66          46
 9 8cents         2 X5             68          44
10 8cents         2 X6             71          51
# i 14 more rows
```

If you check this, you'll see that `replicate` gets turned into the same thing in the longer dataframe that we had earlier, so you can do it either way.

The moral of the story is that when you are planning to do a pivot-wider, you ought to devote some attention to which *rows* things are going into. Sometimes you can get away with just doing it and it works, but thinking about rows is how to diagnose it when it doesn't. (The ideas below also appear in Lecture 14a.) Here's another mini-example where the observations are matched pairs but they come to us long, like two-sample data:

```r
d <- tribble(
~obs, ~y, ~time,
1, 10, "pre",
2, 13, "post",
3, 12, "pre",
4, 14, "post",
5, 13, "pre",
6, 15, "post"
)
d %>% pivot_wider(names_from = time, values_from = y)
```

```
# A tibble: 6 x 3
    obs   pre  post
  <dbl> <dbl> <dbl>
1     1    10    NA
2     2    NA    13
3     3    12    NA
4     4    NA    14
5     5    13    NA
6     6    NA    15
```

Oh. The columns are all right, but the rows certainly are not.

The problem is that the only thing left after `y` and `time` have been used in the `pivot_wider` is the column `obs`, and there are six values there, so there are six rows. This is, in a way, the opposite of the problem we had before; now, there is *not enough* data to fill the twelve cells of the wider dataframe. For example, there is no `pre` measurement in the row where `obs` is 2, so this cell of the wider dataframe is empty: it has a missing value in it.

The problem is that the `obs` column numbered the six observations 1 through 6, but really they are three groups of two observations on three people, so instead of `obs` we need a column called `person` that shows which observations are the matched pairs, like this:

```r
d <- tribble(
~person, ~y, ~time,
1, 10, "pre",
1, 13, "post",
2, 12, "pre",
2, 14, "post",
3, 13, "pre",
3, 15, "post"
)
```

Now there are going to be three rows with a pre and a post in each:

```
d %>% pivot_wider(names_from = time, values_from = y)
```

```
# A tibble: 3 x 3
  person   pre  post
   <dbl> <dbl> <dbl>
1      1    10    13
2      2    12    14
3      3    13    15
```

`pivot_wider` requires more thinking than `pivot_longer`, and when it does something mysterious, that's when you need to have some understanding of how it works, so that you can fix things up.

■

(c) Make a suitable graph of temperature, consumption and price in your tidy dataframe. Add smooth trends if appropriate. If you were unable to get the data tidy, use my tidy version here. (If you need the other file, right-click on "here" and Copy Link Address.)

Solution

I said earlier to treat price (rather badly labelled as `X1`) as categorical, so we have two quantitative variables and one categorical. This suggests a scatterplot with the two prices distinguished by colours. (We don't have a mechanism for making three-dimensional plots, and in any case if you have a quantitative variable with not that many distinct different values, you can often treat that as categorical, such as price here.)

Before we make a graph, though, we should rename `X1`. The way you might think of is to create a new column with the same values as `X1`, but a new name.[30] Like this. Consumption is the outcome, so it goes on the y-axis:

```
utils %>%
mutate(price = X1) %>%
ggplot(aes(x = temperature, y = consumption, colour = price)) +
geom_point() + geom_smooth()
```

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

[30]This actually creates a copy of the original column, so if you look you now have two columns with the same thing in them, one with a bad name and one with a good one.

I said smooth trends rather than lines, because you don't know until you draw the graph whether the trends *are* lines. If they're not, there's not much point in drawing lines through them. These ones are rather clearly curves, which we take up in the next part.

If you fail to rename `X1`, that's what will appear on the legend, and the first thing your reader would ask is "what is `X1`?" When writing, you need to think of your reader, since they are (in the real world) paying you for your work.

Extra: there is an official `rename` also. I haven't used that in class, so if you discover this, make sure to say where you found out about it from:

```
utils %>%
rename(price = X1)
```

```
# A tibble: 24 x 4
   price  col   temperature consumption
   <chr>  <chr>       <dbl>       <dbl>
 1 8cents X3             31          55
 2 8cents X3             62          41
 3 8cents X4             34          49
 4 8cents X4             66          46
 5 8cents X5             39          46
 6 8cents X5             68          44
```

```
 7 8cents X6             42          47
 8 8cents X6             71          51
 9 8cents X7             47          40
10 8cents X7             75          62
# i 14 more rows
```

The syntax is "new name equals old one". I used to think it was something like "take the column called `X1` and rename it to `price`", but as you see, that's exactly backwards. The English-language version is "create a new column called `price` from the column previously called `X1`".

■

(d) What patterns or trends do you see on your graph? Do they make practical sense? There are two things I would like you to comment on.

Solution

The two things are:

- the relationships are both curves, going down and then up again.
- the blue curve is above the red one.

If the temperature is low (30 degrees F is just below freezing), people will need to heat their houses, and the electricity consumption to do this is reflected in the curves being higher at the left. (Not all people have electric heating, but at least some people do, and electric heating uses a lot of electricity.) When the temperature is high, people will turn on the air-conditioning (which is usually electric), and that explains the sharp increase in consumption at high temperatures. In between is the zone where the house stays a good temperature without heating or cooling.

So why is the blue curve above the red one? This is saying that when electricity is cheaper, people will use more of it. (This seems to be particularly true when the temperature is high; people might "crank" the air-conditioning if it doesn't cost too much to run.) Conversely, if electricity is more expensive, people might be more thoughtful about what temperature to turn on the heating or AC. (For example, in summer you might keep the drapes closed so that it stays cooler inside without needing to use the AC so much.)

■

15.26 Tidy blood pressure

Going to the dentist is scary for a lot of people. One way in which this might show up is that people might have higher blood pressure on average before their dentist's appointment than an hour after the appointment is done. Ten randomly-chosen individuals have their (systolic[31]) blood pressure measured while they are in a dentist's waiting room, and then again one hour after their appointment is finished.

You might have seen a tidy version of this data set before.

The data as I originally received it is in http://ritsokiguess.site/datafiles/blood_pressure2.csv.

(a) Read in and display the data as originally received.

Solution

You ought to be suspicious that something is going to be up with the layout. With that in mind, I'm using a "disposable" name for this dataframe:

```
my_url <- "http://ritsokiguess.site/datafiles/blood_pressure2.csv"
bp0 <- read_csv(my_url)
```

```
Rows: 2 Columns: 11
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (1): time
dbl (10): p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bp0
```

```
# A tibble: 2 x 11
  time      p1    p2    p3    p4    p5    p6    p7    p8    p9   p10
  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 before   132   135   149   133   119   121   128   132   119   110
2 after    118   137   140   139   107   116   122   124   115   103
```

[31] A blood pressure is usually given as two numbers, like "120 over 80". The first number, which is the one shown in our data, is called the systolic blood pressure. It is the pressure in the arteries when the heart is pumping. The second is called the diastolic blood pressure, and it is the pressure in the arteries when the heart is resting.

■

(b) Describe briefly how the data you read in is not tidy, bearing in mind how the data were collected and how they would be analysed.

Solution

In short, the things that are rows should be columns, and the things that are columns should be rows. Or, the individuals (people, here) should be in rows but they are in columns. Or, the variables (time points) should be in columns but they are in rows. One of those.

Another way to go at it is to note that the numbers are all blood pressure measurements, and so should be all in one column, labelled by which time and individual they belong to. This is, however, not quite right, for reasons of how the data were collected. They are pairs of measurements on the same individual, and so there should be (for something like a matched pairs t-test) a column of before measurements and a column of after measurements. This will mean some extra work in the next part to get it tidy.

■

(c) Produce a tidy dataframe from the one you read in from the file. (How many rows should you have?)

Solution

The usual starting point for these is to get all the measurements into one column and see what to do after that. This is `pivot_longer`:

```
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp")
```

```
# A tibble: 20 x 3
   time   person    bp
   <chr>  <chr>  <dbl>
 1 before p1       132
 2 before p2       135
 3 before p3       149
 4 before p4       133
 5 before p5       119
 6 before p6       121
 7 before p7       128
 8 before p8       132
 9 before p9       119
10 before p10      110
11 after  p1       118
12 after  p2       137
```

```
13 after  p3       140
14 after  p4       139
15 after  p5       107
16 after  p6       116
17 after  p7       122
18 after  p8       124
19 after  p9       115
20 after  p10      103
```

This would be tidy if we had 20 independent observations from 20 different people. But we don't. We only have 10 people, with two measurements on each, so we should only have 10 rows. Having made things longer, they are now *too* long, and we have to make it wider again.

We want to have a column of before and a column of after, so the names of the new columns are coming from what I called `time`. The values are coming from what I called `bp`, so, gluing the `pivot_wider` on the end of the pipe:

```
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp") %>%
  pivot_wider(names_from = time, values_from = bp) -> blood_pressure
blood_pressure
```

```
# A tibble: 10 x 3
   person before after
   <chr>   <dbl> <dbl>
 1 p1        132   118
 2 p2        135   137
 3 p3        149   140
 4 p4        133   139
 5 p5        119   107
 6 p6        121   116
 7 p7        128   122
 8 p8        132   124
 9 p9        119   115
10 p10       110   103
```

This is now tidy, so I gave it a "permanent" name.

I laid out the steps of my thinking, so you could follow my logic. I'm expecting your *thinking* to be about the same as mine, but the work you hand in can certainly be the finished pipe I had just above, as if you thought of it right away.

Extra: `pivot_wider` is smarter than you think, but it can be helpful to know what it does, in order to help diagnose when things go wrong. Let's go back and look at the too-long dataframe again:

```
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp")
```

```
# A tibble: 20 x 3
   time   person    bp
   <chr>  <chr>  <dbl>
 1 before p1       132
 2 before p2       135
 3 before p3       149
 4 before p4       133
 5 before p5       119
 6 before p6       121
 7 before p7       128
 8 before p8       132
 9 before p9       119
10 before p10      110
11 after  p1       118
12 after  p2       137
13 after  p3       140
14 after  p4       139
15 after  p5       107
16 after  p6       116
17 after  p7       122
18 after  p8       124
19 after  p9       115
20 after  p10      103
```

Each one of those values in `bp` has to go somewhere in the wider dataframe. In particular, it needs to go in a particular row and column. The column is pretty obvious: the column whose name is in the `time` column. But the row is much less obvious. How does `pivot_wider` figure it out? Well, it looks for all combinations of values in the *other* columns, the ones not mentioned in the `pivot_wider`, and makes a row for each of those. In this case, the only other column is `person`, so it makes one row for each person. Since there is one before and one after measurement for each person, everything works smoothly.

This enables us to try a couple of what-ifs to see what can go wrong.

First, what if there's no person column at all, so there is nothing to say what row an observation should go in?

```r
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp") %>%
select(-person) %>%
pivot_wider(names_from = time, values_from = bp)
```

```
Warning: Values from `bp` are not uniquely identified; output will contain list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} %>%
  dplyr::group_by(time) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
# A tibble: 1 x 2
  before     after
  <list>     <list>
1 <dbl [10]> <dbl [10]>
```

It kinda works, but with a warning. The warning says "values are not uniquely identified", which is a posh way to say that it doesn't know where to put them (because there is no longer a way to say which row each observation should go in).

Here's another one, similar:

```r
d <- tribble(
~g, ~id, ~y,
"a", "p1", 10,
"a", "p2", 11,
"b", "p1", 12,
"b", "p2", 13,
"a", "p3", 14,
"a", "p1", 15
)
d
```

```
# A tibble: 6 x 3
  g     id        y
  <chr> <chr> <dbl>
1 a     p1       10
2 a     p2       11
3 b     p1       12
```

```
4 b     p2       13
5 a     p3       14
6 a     p1       15
```

When we do this:

```
d %>% pivot_wider(names_from = g, values_from = y)
```

```
Warning: Values from `y` are not uniquely identified; output will contain list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} %>%
  dplyr::group_by(id, g) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
# A tibble: 3 x 3
  id    a         b
  <chr> <list>    <list>
1 p1    <dbl [2]> <dbl [1]>
2 p2    <dbl [1]> <dbl [1]>
3 p3    <dbl [1]> <NULL>
```

we get list-columns again (and the same warning). What this output is telling you is that mostly there is one number per id-group combination (the `dbl[1]`) but there are two observations labelled id `p1` and group `a`, and no observations at all labelled id `p3` and group `b`. It turns out^[I know because I made these data up. that the last row of the tribble contains errors. Fix them, and all is good:

```
d <- tribble(
~g, ~id, ~y,
"a", "p1", 10,
"a", "p2", 11,
"b", "p1", 12,
"b", "p2", 13,
"a", "p3", 14,
"b", "p3", 15
)
d %>% pivot_wider(names_from = g, values_from = y)
```

```
# A tibble: 3 x 3
  id        a     b
  <chr> <dbl> <dbl>
1 p1       10    12
2 p2       11    13
3 p3       14    15
```

One last one:

```
d <- tribble(
~id, ~g, ~y,
1, "a", 10,
2, "a", 11,
3, "a", 12,
4, "b", 13,
5, "b", 14,
6, "b", 15
)
d
```

```
# A tibble: 6 x 3
     id g         y
  <dbl> <chr> <dbl>
1     1 a        10
2     2 a        11
3     3 a        12
4     4 b        13
5     5 b        14
6     6 b        15
```

and then

```
d %>% pivot_wider(names_from = g, values_from = y)
```

```
# A tibble: 6 x 3
     id     a     b
  <dbl> <dbl> <dbl>
1     1    10    NA
2     2    11    NA
3     3    12    NA
4     4    NA    13
```

```
5     5    NA    14
6     6    NA    15
```

Where did those missing values come from? If you go back and look at this `d`, you'll see that each person has only *one* measurement, *either* an `a` or a `b`, not both. There is, for example, nothing to go in the `a` column for person number 4, because their only measurement was in group `b`. This kind of thing happens with two independent samples, and is a warning that you don't *need* to pivot wider; it's already tidy:

```
d
```

```
# A tibble: 6 x 3
     id g         y
  <dbl> <chr> <dbl>
1     1 a        10
2     2 a        11
3     3 a        12
4     4 b        13
5     5 b        14
6     6 b        15
```

Think about the kind of layout you need for a two-sample t-test.

■

(d) What kind of test might you run on these data? Explain briefly.

Solution

This is a matched-pairs experiment, so it needs a matched-pairs analysis. This could be a matched-pairs t-test, or a sign test on the differences (testing that the population median difference is zero). You can suggest either, since we haven't drawn any graphs yet, but "sign test" is not enough; you need to say something about what kind of sign test. (It's actually quicker to answer "matched-pairs t-test" since you don't need any more detail than that.)

■

(e) Draw a suitable graph of these data.

Solution

Given that we are going to do a matched-pairs analysis of some kind, the best graph looks at the *differences* between the two measurements. So calculate them first, and then make a one-sample plot of them, such as a histogram:

```
blood_pressure %>% mutate(diff = before - after) %>%
ggplot(aes(x=diff)) + geom_histogram(bins=5)
```

You will need a suitably small number of bins, since we only have ten observations. You can take the differences the other way around if you prefer; they will then be mostly negative.

You might have looked at the two quantitative columns and thought "scatterplot":

```
ggplot(blood_pressure, aes(x=before, y=after)) + geom_point()
```

This says that if the blood pressure before was large, the blood pressure afterwards is as well. This is fair enough, but it is the answer to a question *we don't care about*. What we *do* care about is whether the after measurement is bigger than the before one *for the same person*, which this graph does not show. So this is not the best.

To rescue this graph, you can add the line $y = x$ to it. The value of this is that a point above this line has the after measurement bigger than the corresponding before one, and a point below the line has the after measurement smaller. You will need to find out how to add a line with a given intercept and slope to the plot, since I haven't shown you how to do it. It's called `geom_abline`, thus:

```
ggplot(blood_pressure, aes(x=before, y=after)) + geom_point() +
geom_abline(intercept = 0, slope = 1)
```

This is insightful, because most of the points are below the line, so that most of the before measurements were bigger than the corresponding after ones.

Note that if you put a *regression line* on your plot, you will need to offer a convincing explanation of why that offers insight, which I think you will find difficult.

Finally, if you thought the long data frame was tidy, this one:

```
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp")
```

```
# A tibble: 20 x 3
   time   person    bp
   <chr>  <chr>  <dbl>
 1 before p1       132
 2 before p2       135
 3 before p3       149
 4 before p4       133
 5 before p5       119
 6 before p6       121
 7 before p7       128
 8 before p8       132
 9 before p9       119
10 before p10      110
11 after  p1       118
```

```
12 after  p2       137
13 after  p3       140
14 after  p4       139
15 after  p5       107
16 after  p6       116
17 after  p7       122
18 after  p8       124
19 after  p9       115
20 after  p10      103
```

then you can rescue some points here by making a suitable plot of that. A boxplot is not enough:

```
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp") %>%
ggplot(aes(x=time, y=bp)) + geom_boxplot()
```

because you have *lost* the connection between the two measurements for each person. To keep that connection, start with the same plot but as points rather than boxes:

```
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp") %>%
ggplot(aes(x=time, y=bp)) + geom_point()
```

and then join the two points that belong to the same person. This is done with `geom_line` as usual, only you have to say which points are going to be joined, namely the two for each person, and to do that, you specify `person` in `group`:

```
bp0 %>% pivot_longer(-time, names_to="person", values_to="bp") %>%
ggplot(aes(x=time, y=bp, group=person)) + geom_point() + geom_line()
```

Most of the lines are going uphill, so most of the `after` measurements are less than the corresponding `before` ones.[32]

■

[32]This is known in the trade as a *spaghetti plot* because the lines resemble strands of spaghetti.

16 Simple regression

```
library(tidyverse)
```

16.1 Rainfall in California

The data in link are rainfall and other measurements for 30 weather stations in California. Our aim is to understand how well the annual rainfall at these stations (measured in inches) can be predicted from the other measurements, which are the altitude (in feet above sea level), the latitude (degrees north of the equator) and the distance from the coast (in miles).

(a) Read the data into R. You'll have to be careful here, since the values are space-delimited, but sometimes by more than one space, to make the columns line up. `read_table2`, with filename or url, will read it in. One of the variables is called `rainfall`, so as long as you *do not* call the data frame that, you should be safe.

(b) Make a boxplot of the rainfall figures, and explain why the values are reasonable. (A rainfall cannot be negative, and it is unusual for a annual rainfall to exceed 60 inches.) A `ggplot` boxplot needs *something* on the x-axis: the number 1 will do.

(c) Plot `rainfall` against each of the other quantitative variables (that is, not `station`).

(d) Look at the relationship of each other variable with `rainfall`. Justify the assertion that `latitude` seems most strongly related with `rainfall`. Is that relationship positive or negative? linear? Explain briefly.

(e) Fit a regression with `rainfall` as the response variable, and `latitude` as your explanatory variable. What are the intercept, slope and R-squared values? Is there a *significant* relationship between `rainfall` and your explanatory variable? What does that mean?

(f) Fit a multiple regression predicting `rainfall` from all three of the other (quantitative) variables. Display the results. Comment is coming up later.

(g) What is the R-squared for the regression of the last part? How does that compare with the R-squared of your regression in part (e)?

(h) What do you conclude about the importance of the variables that you did *not* include in your model in (e)? Explain briefly.

(i) Make a suitable hypothesis test that the variables `altitude` and `fromcoast` significantly improve the prediction of `rainfall` over the use of `latitude` alone. What do you conclude?

16.2 Carbon monoxide in cigarettes

The (US) Federal Trade Commission assesses cigarettes according to their tar, nicotine and carbon monoxide contents. In a particular year, 25 brands were assessed. For each brand, the tar, nicotine and carbon monoxide (all in milligrams) were measured, along with the weight in grams. Our aim is to predict carbon monoxide from any or all of the other variables. The data are in link. These are aligned by column (except for the variable names), with more than one space between each column of data.

(a) Read the data into R, and check that you have 25 observations and 4 variables.

(b) Run a regression to predict carbon monoxide from the other variables, and obtain a summary of the output.

(c) Which one of your explanatory variables would you remove from this regression? Explain (very) briefly. Go ahead and fit the regression without it, and describe how the change in R-squared from the regression in (b) was entirely predictable.

(d) Fit a regression predicting carbon monoxide from `nicotine` *only*, and display the summary.

(e) `nicotine` was far from being significant in the model of (c), and yet in the model of (d), it was *strongly* significant, and the R-squared value of (d) was almost as high as that of (c). What does this say about the importance of `nicotine` as an explanatory variable? Explain, as briefly as you can manage.

(f) Make a "pairs plot": that is, scatter plots between all pairs of variables. This can be done by feeding the whole data frame into `plot`.[1] Do you see any strong relationships that do *not* include `co`? Does that shed any light on the last part? Explain briefly (or "at length" if that's how it comes out).

16.3 Maximal oxygen uptake in young boys

A physiologist wanted to understand the relationship between physical characteristics of pre-adolescent boys and their maximal oxygen uptake (millilitres of oxygen per kilogram of body weight). The data are in link for a random sample of 10 pre-adolescent boys. The variables are (with units):

[1] This is a base graphics graph rather than a `ggplot` one, but it will do for our purposes.

- `uptake`: Oxygen uptake (millitres of oxygen per kilogram of body weight)
- `age`: boy's age (years)
- `height`: boy's height (cm)
- `weight`: boy's weight (kg)
- `chest`: chest depth (cm).

(a) Read the data into R and confirm that you do indeed have 10 observations.

(b) Fit a regression predicting oxygen uptake from all the other variables, and display the results.

(c) (A one-mark question.) Would you say, on the evidence so far, that the regression fits well or badly? Explain (very) briefly.

(d) It seems reasonable that an older boy should have a greater oxygen uptake, all else being equal. Is this supported by your output? Explain briefly.

(e) It seems reasonable that a boy with larger weight should have larger lungs and thus a *statistically significantly* larger oxygen uptake. Is that what happens here? Explain briefly.

(f) Fit a model that contains only the significant explanatory variables from your first regression. How do the R-squared values from the two regressions compare? (The last sentence asks for more or less the same thing as the next part. Answer it either here or there. Either place is good.)

(g) How has R-squared changed between your two regressions? Describe what you see in a few words.

(h) Carry out a test comparing the fit of your two regression models. What do you conclude, and therefore what recommendation would you make about the regression that would be preferred?

(i) Obtain a table of correlations between all the variables in the data frame. Do this by feeding the whole data frame into `cor`. We found that a regression predicting oxygen uptake from just `height` was acceptably good. What does your table of correlations say about why that is? (Hint: look for all the correlations that are *large*.)

16.4 Facebook friends and grey matter

Is there a relationship between the number of Facebook friends a person has, and the density of grey matter in the areas of the brain associated with social perception and associative memory? To find out, a 2012 study measured both of these variables for a sample of 40 students at City University in London (England). The data are at link. The grey matter density is on a z-score standardized scale. The values are separated by *tabs*.

The aim of this question is to produce an R Markdown report that contains your answers to the questions below.

You should aim to make your report flow smoothly, so that it would be pleasant for a grader to read, and can stand on its own as an analysis (rather than just being the answer to a question that I set you). Some suggestions: give your report a title and arrange it into sections with an Introduction; add a small amount of additional text here and there explaining what you are doing and why. I don't expect you to spend a large amount of time on this, but I *do* hope you will make some effort. (My report came out to 4 Word pages.)

(a) Read in the data and make a scatterplot for predicting the number of Facebook friends from the grey matter density. On your scatterplot, add a smooth trend.

(b) Describe what you see on your scatterplot: is there a trend, and if so, what kind of trend is it? (Don't get too taken in by the exact shape of your smooth trend.) Think "form, direction, strength".

(c) Fit a regression predicting the number of Facebook friends from the grey matter density, and display the output.

(d) Is the slope of your regression line significantly different from zero? What does that mean, in the context of the data?

(e) Are you surprised by the results of parts (b) and (d)? Explain briefly.

(f) Obtain a scatterplot with the regression line on it.

(g) Obtain a plot of the residuals from the regression against the fitted values, and comment briefly on it.

16.5 Endogenous nitrogen excretion in carp

A paper in Fisheries Science reported on variables that affect "endogenous nitrogen excretion" or ENE in carp raised in Japan. A number of carp were divided into groups based on body weight, and each group was placed in a different tank. The mean body weight of the carp placed in each tank was recorded. The carp were then fed a protein-free diet three times daily for a period of 20 days. At the end of the experiment, the amount of ENE in each tank was

measured, in milligrams of total fish body weight per day. (Thus it should not matter that some of the tanks had more fish than others, because the scaling is done properly.)

For this question, write a report in R Markdown that answers the questions below and contains some narrative that describes your analysis. Create an HTML document from your R Markdown.

(a) Read the data in from link. There are 10 tanks.

(b) Create a scatterplot of ENE (response) against bodyweight (explanatory). Add a smooth trend to your plot.

(c) Is there an upward or downward trend (or neither)? Is the relationship a line or a curve? Explain briefly.

(d) Fit a straight line to the data, and obtain the R-squared for the regression.

(e) Obtain a residual plot (residuals against fitted values) for this regression. Do you see any problems? If so, what does that tell you about the relationship in the data?

(f) Fit a parabola to the data (that is, including an x-squared term). Compare the R-squared values for the models in this part and part (d). Does that suggest that the parabola model is an improvement here over the linear model?

(g) Is the test for the slope coefficient for the squared term significant? What does this mean?

(h) Make the scatterplot of part (b), but add the fitted curve. Describe any way in which the curve fails to fit well.

(i) Obtain a residual plot for the parabola model. Do you see any problems with it? (If you do, I'm not asking you to do anything about them in this question, but I will.)

16.6 Salaries of social workers

Another salary-prediction question: does the number of years of work experience that a social worker has help to predict their salary? Data for 50 social workers are in link.

(a) Read the data into R. Check that you have 50 observations on two variables. Also do something to check that the years of experience and annual salary figures look reasonable overall.

(b) Make a scatterplot showing how salary depends on experience. Does the nature of the trend make sense?

(c) Fit a regression predicting salary from experience, and display the results. Is the slope positive or negative? Does that make sense?

(d) Obtain and plot the residuals against the fitted values. What problem do you see?

(e) The problem you unearthed in the previous part is often helped by a transformation. Run Box-Cox on your data to find a suitable transformation. What transformation is suggested?

(f) Calculate a new variable as suggested by your transformation. Use your transformed response in a regression, showing the summary.

(g) Obtain and plot the residuals against the fitted values for this regression. Do you seem to have solved the problem with the previous residual plot?

16.7 Predicting volume of wood in pine trees

In forestry, the financial value of a tree is the volume of wood that it contains. This is difficult to estimate while the tree is still standing, but the diameter is easy to measure with a tape measure (to measure the circumference) and a calculation involving π, assuming that the cross-section of the tree is at least approximately circular. The standard measurement is "diameter at breast height" (that is, at the height of a human breast or chest), defined as being 4.5 feet above the ground.

Several pine trees had their diameter measured shortly before being cut down, and for each tree, the volume of wood was recorded. The data are in link. The diameter is in inches and the volume is in cubic inches. Is it possible to predict the volume of wood from the diameter?

(a) Read the data into R and display the values (there are not very many).

(b) Make a suitable plot.

(c) Describe what you learn from your plot about the relationship between diameter and volume, if anything.

(d) Fit a (linear) regression, predicting volume from diameter, and obtain the `summary`. How would you describe the R-squared?

(e) Draw a graph that will help you decide whether you trust the linearity of this regression. What do you conclude? Explain briefly.

(f) What would you guess would be the volume of a tree of diameter zero? Is that what the regression predicts? Explain briefly.

(g) A simple way of modelling a tree's shape is to pretend it is a cone, like this, but probably taller and skinnier:

with its base on the ground. What is the relationship between the *diameter* (at the base) and volume of a cone? (If you don't remember, look it up. You'll probably get a formula in terms of the radius, which you'll have to convert. Cite the website you used.)

(h) Fit a regression model that predicts volume from diameter according to the formula you obtained in the previous part. You can assume that the trees in this data set are of similar heights, so that the height can be treated as a constant.
Display the results.

16.8 Tortoise shells and eggs

A biologist measured the length of the carapace (shell) of female tortoises, and then x-rayed the tortoises to count how many eggs they were carrying. The length is measured in millimetres. The data are in link. The biologist is wondering what kind of relationship, if any, there is between the carapace length (as an explanatory variable) and the number of eggs (as a response variable).

(a) Read in the data, and check that your values look reasonable.

(b) Obtain a scatterplot, with a smooth trend, of the data.

(c) The biologist expected that a larger tortoise would be able to carry more eggs. Is that what the scatterplot is suggesting? Explain briefly why or why not.

(d) Fit a straight-line relationship and display the summary.

(e) Add a squared term to your regression, fit that and display the summary.

(f) Is a curve better than a line for these data? Justify your answer in two ways: by comparing a measure of fit, and by doing a suitable test of significance.

(g) Make a residual plot for the straight line model: that is, plot the residuals against the fitted values. Does this echo your conclusions of the previous part? In what way? Explain briefly.

16.9 Roller coasters

A poll on the Discovery Channel asked people to nominate the best roller-coasters in the United States. We will examine the 10 roller-coasters that received the most votes. Two features of a roller-coaster that are of interest are the distance it drops from start to finish, measured here in feet[2] and the duration of the ride, measured in seconds. Is it true that roller-coasters with a bigger drop also tend to have a longer ride? The data are at link.[3]

(a) Read the data into R and verify that you have a sensible number of rows and columns.

(b) Make a scatterplot of duration (response) against drop (explanatory), labelling each roller-coaster with its name in such a way that the labels do not overlap. Add a regression line to your plot.

(c) Would you say that roller-coasters with a larger drop tend to have a longer ride? Explain briefly.

(d) Find a roller-coaster that is unusual compared to the others. What about its combination of `drop` and `duration` is unusual?

16.10 Running and blood sugar

A diabetic wants to know how aerobic exercise affects his blood sugar. When his blood sugar reaches 170 (mg/dl), he goes out for a run at a pace of 10 minutes per mile. He runs different distances on different days. Each time he runs, he measures his blood sugar after the run. (The preferred blood sugar level is between 80 and 120 on this scale.) The data are in the file link. Our aim is to predict blood sugar from distance.

[2] Roller-coasters work by gravity, so there must be some drop.

[3] These are not to be confused with what your mom insists that you place between your coffee mug and the table.

(a) Read in the data and display the data frame that you read in.

(b) Make a scatterplot and add a smooth trend to it.

(c) Would you say that the relationship between blood sugar and running distance is approximately linear, or not? It is therefore reasonable to use a regression of blood sugar on distance? Explain briefly.

(d) Fit a suitable regression, and obtain the regression output.

(e) How would you *interpret* the slope? That is, what is the slope, and what does that mean about blood sugar and running distance?

(f) Is there a (statistically) significant relationship between running distance and blood sugar? How do you know? Do you find this surprising, given what you have seen so far? Explain briefly.

(g) This diabetic is planning to go for a 3-mile run tomorrow and a 5-mile run the day after. Obtain suitable 95% intervals that say what his blood sugar might be after each of these runs.

(h) Which of your two intervals is longer? Does this make sense? Explain briefly.

16.11 Calories and fat in pizza

The file at link came from a spreadsheet of information about 24 brands of pizza: specifically, per 5-ounce serving, the number of calories, the grams of fat, and the cost (in US dollars). The names of the pizza brands are quite long. This file may open in a spreadsheet when you browse to the link, depending on your computer's setup.

(a) Read in the data and display at least some of the data frame. Are the variables of the right types? (In particular, why is the number of calories labelled one way and the cost labelled a different way?)

(b) Make a scatterplot for predicting calories from the number of grams of fat. Add a smooth trend. What kind of relationship do you see, if any?

(c) Fit a straight-line relationship, and display the intercept, slope, R-squared, etc. Is there a real relationship between the two variables, or is any apparent trend just chance?

(d) Obtain a plot of the residuals against the fitted values for this regression. Does this indicate that there are any problems with this regression, or not? Explain briefly.

(e) The research assistant in this study returns with two new brands of pizza (ones that were not in the original data). The fat content of a 5-ounce serving was 12 grams for the first brand and 20 grams for the second brand. For each of these brands of pizza, obtain a suitable 95% interval for the number of calories contained in a 5-ounce serving.

16.12 Where should the fire stations be?

In city planning, one major issue is where to locate fire stations. If a city has too many fire stations, it will spend too much on running them, but if it has too few, there may be unnecessary fire damage because the fire trucks take too long to get to the fire.

The first part of a study of this kind of issue is to understand the relationship between the distance from the fire station (measured in miles in our data set) and the amount of fire damage caused (measured in thousands of dollars). A city recorded the fire damage and distance from fire station for 15 residential fires (which you can take as a sample of "all possible residential fires in that city"). The data are in link.

(a) Read in and display the data, verifying that you have the right number of rows and the right columns.

(b) * Obtain a 95% confidence interval for the mean fire damage. (There is nothing here from STAD29, and your answer should have nothing to do with distance.)

(c) Draw a scatterplot for predicting the amount of fire damage from the distance from the fire station. Add a smooth trend to your plot.

(d) * Is there a relationship between distance from fire station and fire damage? Is it linear or definitely curved? How strong is it? Explain briefly.

(e) Fit a regression predicting fire damage from distance. How is the R-squared consistent (or inconsistent) with your answer from part~(here)?

(f) *Obtain a 95% confidence interval for the mean fire damage* for a residence that is 4 miles from the nearest fire station*. (Note the contrast with part~(here).)

(g) Compare the confidence intervals of parts (here) and (here). Specifically, compare their centres and their lengths, and explain briefly why the results make sense.

16.13 Making it stop

If you are driving, and you hit the brakes, how far do you travel before coming to a complete stop? Presumably this depends on how fast you are going. Knowing this relationship is important in setting speed limits on roads. For example, on a very bendy road, the speed limit needs to be low, because you cannot see very far ahead, and there could be something just out of sight that you need to stop for.

Data were collected for a typical car and driver, as shown in http://ritsokiguess.site/datafiles/stopping.csv. These are American data, so the speeds are miles per hour and the stopping distances are in feet.

(a) Read in and display (probably all of) the data.

(b) Make a suitable plot of the data.

(c) Describe any trend you see in your graph.

(d) Fit a linear regression predicting stopping distance from speed. (You might have some misgivings about doing this, but do it anyway.)

(e) Plot the residuals against the fitted values for this regression.

(f) What do you learn from the residual plot? Does that surprise you? Explain briefly.

(g) What is the actual relationship between stopping distance and speed, according to the physics? See if you can find out. Cite any books or websites that you use: that is, include a link to a website, or give enough information about a book that the grader could find it.

(h) Fit the relationship that your research indicated (in the previous part) and display the results. Comment briefly on the R-squared value.

(i) Somebody says to you "if you have a regression with a high R-squared, like 95%, there is no need to look for a better model." How do you respond to this? Explain briefly.

16.14 Predicting height from foot length

Is it possible to estimate the height of a person from the length of their foot? To find out, 33 (male) students had their height and foot length measured. The data are in http://ritsokiguess.site/datafiles/heightfoot.csv.

(a) Read in and display (some of) the data. (If you are having trouble, make sure you have *exactly* the right URL. The correct URL has no spaces or other strange characters in it.)

(b) Make a suitable plot of the two variables in the data frame.

(c) Are there any observations not on the trend of the other points? What is unusual about those observations?

(d) Fit a regression predicting height from foot length, *including* any observations that you identified in the previous part. For that regression, plot the residuals against the fitted values and make a normal quantile plot of the residuals.

(e) Earlier, you identified one or more observations that were off the trend. How does this point or points show up on each of the plots you drew in the previous part?

(f) Any data points that concerned you earlier were actually errors. Create and save a new data frame that does not contain any of those data points.

(g) Run a regression predicting height from foot length for your data set without errors. Obtain a plot of the residuals against fitted values and a normal quantile plot of the residuals for this regression.

(h) Do you see any problems on the plots you drew in the previous part? Explain briefly.

(i) Find a way to plot the data and *both* regression lines on the same plot, in such a way that you can see which regression line is which. If you get help from anything outside the course materials, cite your source(s).

(j) Discuss briefly how removing the observation(s) that were errors has changed where the regression line goes, and whether that is what you expected.

My solutions follow:

16.15 Rainfall in California

The data in link are rainfall and other measurements for 30 weather stations in California. Our aim is to understand how well the annual rainfall at these stations (measured in inches) can be predicted from the other measurements, which are the altitude (in feet above sea level), the latitude (degrees north of the equator) and the distance from the coast (in miles).

(a) Read the data into R. You'll have to be careful here, since the values are space-delimited, but sometimes by more than one space, to make the columns line up. `read_table2`, with filename or url, will read it in. One of the variables is called `rainfall`, so as long as you *do not* call the data frame that, you should be safe.

Solution

I used `rains` as the name of my data frame:

```
my_url="http://ritsokiguess.site/datafiles/calirain.txt"
rains=read_table2(my_url)
```

```
Warning: `read_table2()` was deprecated in readr 2.0.0.
i Please use `read_table()` instead.
```

```
-- Column specification --------------------------------------------------------
cols(
  station = col_character(),
  rainfall = col_double(),
  altitude = col_double(),
```

```
  latitude = col_double(),
  fromcoast = col_double()
)
```

I have the right number of rows and columns.

There is also `read_table`, but that requires *all* the columns, including the header row, to be lined up. You can try that here and see how it fails.

I don't need you to investigate the data yet (that happens in the next part), but this is interesting (to me):

```
rains
```

```
# A tibble: 30 x 5
   station      rainfall altitude latitude fromcoast
   <chr>           <dbl>    <dbl>    <dbl>     <dbl>
 1 Eureka           39.6       43     40.8         1
 2 RedBluff         23.3      341     40.2        97
 3 Thermal          18.2     4152     33.8        70
 4 FortBragg        37.5       74     39.4         1
 5 SodaSprings      49.3     6752     39.3       150
 6 SanFrancisco     21.8       52     37.8         5
 7 Sacramento       18.1       25     38.5        80
 8 SanJose          14.2       95     37.4        28
 9 GiantForest      42.6     6360     36.6       145
10 Salinas          13.8       74     36.7        12
# i 20 more rows
```

Some of the station names are two words, but they have been smooshed into one word, so that `read_table2` will recognize them as a single thing. Someone had already done that for us, so I didn't even have to do it myself.

If the station names had been two genuine words, a `.csv` would probably have been the best choice (the actual data values being separated by commas then, and not spaces).

■

(b) Make a boxplot of the rainfall figures, and explain why the values are reasonable. (A rainfall cannot be negative, and it is unusual for a annual rainfall to exceed 60 inches.) A `ggplot` boxplot needs *something* on the x-axis: the number 1 will do.

Solution

```
ggplot(rains,aes(y=rainfall,x=1))+geom_boxplot()
```

There is only one rainfall over 60 inches, and the smallest one is close to zero but positive, so that is good.

Another possible plot here is a histogram, since there is only one quantitative variable:

```
ggplot(rains, aes(x=rainfall))+geom_histogram(bins=7)
```

This clearly shows the rainfall value above 60 inches, but some other things are less clear: are those two rainfall values around 50 inches above or below 50, and are those six rainfall values near zero actually above zero? Extra: What stations have those extreme values? Should you wish to find out:

```
rains %>% filter(rainfall>60)
```

```
# A tibble: 1 x 5
  station      rainfall altitude latitude fromcoast
  <chr>           <dbl>    <dbl>    <dbl>     <dbl>
1 CrescentCity     74.9       35     41.7         1
```

This is a place right on the Pacific coast, almost up into Oregon (it's almost the northernmost of all the stations). So it makes sense that it would have a high rainfall, if anywhere does. (If you know anything about rainy places, you'll probably think of Vancouver and Seattle, in the Pacific Northwest.) Here it is: link. Which station has less than 2 inches of annual rainfall?

```
rains %>% filter(rainfall<2)
```

```
# A tibble: 1 x 5
  station     rainfall altitude latitude fromcoast
  <chr>          <dbl>    <dbl>    <dbl>     <dbl>
```

```
1 DeathValley     1.66     -178     36.5       194
```

The name of the station is a clue: this one is in the desert. So you'd expect very little rain. Its altitude is *negative*, so it's actually *below* sea level. This is correct. Here is where it is: link.

■

(c) Plot `rainfall` against each of the other quantitative variables (that is, not `station`).

Solution

That is, `altitude`, `latitude` and `fromcoast`. The obvious way to do this (perfectly acceptable) is one plot at a time:

```
ggplot(rains,aes(y=rainfall,x=altitude))+geom_point()
```

```
ggplot(rains,aes(y=rainfall,x=latitude))+geom_point()
```

and finally

```
ggplot(rains,aes(y=rainfall,x=fromcoast))+geom_point()
```

You can add a smooth trend to these if you want. Up to you. Just the points is fine with me.

Here is a funky way to get all three plots in one shot:

```
rains %>%
  pivot_longer(altitude:fromcoast, names_to="xname",values_to="x") %>%
  ggplot(aes(x=x,y=rainfall))+geom_point()+
  facet_wrap(~xname,scales="free")
```

This always seems extraordinarily strange if you haven't run into it before. The strategy is to put *all* the x-variables you want to plot into *one* column and then plot your y against the x-column. Thus: make a column of all the x's glued together, labelled by which x they are, then plot y against x but make a different sub-plot or "facet" for each different x-name. The last thing is that each x is measured on a different scale, and unless we take steps, all the sub-plots will have the *same* scale on each axis, which we don't want.

I'm not sure I like how it came out, with three very tall plots. `facet_wrap` can also take an `nrow` or an `ncol`, which tells it how many rows or columns to use for the display. Here, for example, two columns because I thought three was too many:

```
rains %>%
  pivot_longer(altitude:fromcoast, names_to="xname",values_to="x") %>%
```

```
ggplot(aes(x=x,y=rainfall))+geom_point()+
facet_wrap(~xname,scales="free",ncol=2)
```

Now, the three plots have come out about square, or at least "landscape", which I like a lot better.

■

(d) Look at the relationship of each other variable with `rainfall`. Justify the assertion that `latitude` seems most strongly related with `rainfall`. Is that relationship positive or negative? linear? Explain briefly.

Solution

Let's look at the three variables in turn:

- `altitude`: not much of anything. The stations near sea level have rainfall all over the place, though the three highest-altitude stations have the three highest rainfalls apart from Crescent City.
- `latitude`: there is a definite upward trend here, in that stations further north (higher latitude) are likely to have a higher rainfall. I'd call this trend linear (or, not obviously curved), though the two most northerly stations have one higher and one much lower rainfall than you'd expect.

- `fromcoast`: this is a weak downward trend, though the trend is spoiled by those three stations about 150 miles from the coast that have more than 40 inches of rainfall.

Out of those, only `latitude` seems to have any meaningful relationship with `rainfall`.

■

(e) Fit a regression with `rainfall` as the response variable, and `latitude` as your explanatory variable. What are the intercept, slope and R-squared values? Is there a *significant* relationship between `rainfall` and your explanatory variable? What does that mean?

Solution

Save your `lm` into a variable, since it will get used again later:

```
rainfall.1=lm(rainfall~latitude,data=rains)
summary(rainfall.1)
```

```
Call:
lm(formula = rainfall ~ latitude, data = rains)

Residuals:
    Min      1Q  Median      3Q     Max 
-27.297  -7.956  -2.103   6.082  38.262 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -113.3028    35.7210  -3.172  0.00366 ** 
latitude       3.5950     0.9623   3.736  0.00085 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.82 on 28 degrees of freedom
Multiple R-squared:  0.3326,	Adjusted R-squared:  0.3088 
F-statistic: 13.96 on 1 and 28 DF,  p-value: 0.0008495
```

My intercept is -113.3, slope is 3.6 and R-squared is 0.33 or 33%. (I want you to pull these numbers out of the output and round them off to something sensible.) The slope is significantly nonzero, its P-value being 0.00085: rainfall really does depend on latitude, although not strongly so.

Extra: Of course, I can easily do the others as well, though you don't have to:

```r
rainfall.2=lm(rainfall~fromcoast,data=rains)
summary(rainfall.2)
```

```
Call:
lm(formula = rainfall ~ fromcoast, data = rains)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.240  -9.431  -6.603   2.871  51.147 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 23.77306    4.61296   5.154 1.82e-05 ***
fromcoast   -0.05039    0.04431  -1.137    0.265    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16.54 on 28 degrees of freedom
Multiple R-squared:  0.04414,	Adjusted R-squared:   0.01 
F-statistic: 1.293 on 1 and 28 DF,  p-value: 0.2651
```

Here, the intercept is 23.8, the slope is -0.05 and R-squared is a dismal 0.04 (4%). This is a way of seeing that this relationship is really weak, and it doesn't even have a curve to the trend or anything that would compensate for this. I looked at the scatterplot again and saw that if it were not for the point bottom right which is furthest from the coast and has almost no rainfall, there would be almost no trend at all. The slope here is not significantly different from zero, with a P-value of 0.265.

Finally:

```r
rainfall.3=lm(rainfall~altitude,data=rains)
summary(rainfall.3)
```

```
Call:
lm(formula = rainfall ~ altitude, data = rains)

Residuals:
    Min      1Q  Median      3Q     Max 
-20.620  -8.479  -2.729   4.555  58.271 
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 16.514799   3.539141   4.666  6.9e-05 ***
altitude     0.002394   0.001428   1.676    0.105
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16.13 on 28 degrees of freedom
Multiple R-squared:  0.09121,   Adjusted R-squared:  0.05875
F-statistic:  2.81 on 1 and 28 DF,  p-value: 0.1048
```

The intercept is 16.5, the slope is 0.002 and the R-squared is 0.09 or 9%, also terrible. The P-value is 0.105, which is not small enough to be significant.

So it looks as if it's only `latitude` that has any impact at all. This is the only explanatory variable with a significantly nonzero slope. On its own, at least.

■

(f) Fit a multiple regression predicting `rainfall` from all three of the other (quantitative) variables. Display the results. Comment is coming up later.

Solution

This, then:

```r
rainfall.4=lm(rainfall~latitude+altitude+fromcoast,data=rains)
summary(rainfall.4)
```

```
Call:
lm(formula = rainfall ~ latitude + altitude + fromcoast, data = rains)

Residuals:
    Min      1Q  Median      3Q     Max
-28.722  -5.603  -0.531   3.510  33.317

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.024e+02  2.921e+01  -3.505 0.001676 **
latitude     3.451e+00  7.949e-01   4.342 0.000191 ***
altitude     4.091e-03  1.218e-03   3.358 0.002431 **
fromcoast   -1.429e-01  3.634e-02  -3.931 0.000559 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.1 on 26 degrees of freedom
Multiple R-squared:  0.6003,    Adjusted R-squared:  0.5542
F-statistic: 13.02 on 3 and 26 DF,  p-value: 2.205e-05
```

■

(g) What is the R-squared for the regression of the last part? How does that compare with the R-squared of your regression in part (e)?

Solution

The R-squared is 0.60 (60%), which is quite a bit bigger than the R-squared of 0.33 (33%) we got back in (e).

■

(h) What do you conclude about the importance of the variables that you did *not* include in your model in (e)? Explain briefly.

Solution

Both variables `altitude` and `fromcoast` are significant in this regression, so they have *something to add* over and above `latitude` when it comes to predicting rainfall, even though (and this seems odd) they have no apparent relationship with `rainfall` on their own. Another way to say this is that the three variables work together as a team to predict rainfall, and together they do much better than any one of them can do by themselves.
This also goes to show that the scatterplots we began with don't get to the heart of multivariable relationships, because they are only looking at the variables two at a time.

■

(i) Make a suitable hypothesis test that the variables `altitude` and `fromcoast` significantly improve the prediction of `rainfall` over the use of `latitude` alone. What do you conclude?

Solution

This calls for `anova`. Feed this two fitted models, smaller (fewer explanatory variables) first. The null hypothesis is that the two models are equally good (so we should go with the smaller); the alternative is that the larger model is better, so that the extra complication is worth it:

```r
anova(rainfall.1,rainfall.4)
```

```
Analysis of Variance Table

Model 1: rainfall ~ latitude
Model 2: rainfall ~ latitude + altitude + fromcoast
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     28 5346.8
2     26 3202.3  2    2144.5 8.7057 0.001276 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The P-value is small, so we reject the null in favour of the alternative: the regression with all three explanatory variables fits better than the one with just `latitude`, so the bigger model is the one we should go with.

If you have studied these things: this one is a "multiple-partial F-test", for testing the combined significance of more than one x but less than all the x's.[4]

■

16.16 Carbon monoxide in cigarettes

The (US) Federal Trade Commission assesses cigarettes according to their tar, nicotine and carbon monoxide contents. In a particular year, 25 brands were assessed. For each brand, the tar, nicotine and carbon monoxide (all in milligrams) were measured, along with the weight in grams. Our aim is to predict carbon monoxide from any or all of the other variables. The data are in link. These are aligned by column (except for the variable names), with more than one space between each column of data.

(a) Read the data into R, and check that you have 25 observations and 4 variables.

Solution

This specification calls for `read_table`:

```r
my_url <- "http://ritsokiguess.site/datafiles/ftccigar.txt"
cigs <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
```

[4] If you had just one x, you'd use a t-test for its slope, and if you were testing all the x's, you'd use the global F-test that appears in the regression output.

```
  tar = col_double(),
  nicotine = col_double(),
  weight = col_double(),
  co = col_double()
)
```

```r
cigs
```

```
# A tibble: 25 x 4
     tar nicotine weight    co
   <dbl>    <dbl>  <dbl> <dbl>
 1  14.1     0.86  0.985  13.6
 2  16       1.06  1.09   16.6
 3  29.8     2.03  1.16   23.5
 4   8       0.67  0.928  10.2
 5   4.1     0.4   0.946   5.4
 6  15       1.04  0.888  15
 7   8.8     0.76  1.03    9
 8  12.4     0.95  0.922  12.3
 9  16.6     1.12  0.937  16.3
10  14.9     1.02  0.886  15.4
# i 15 more rows
```

Yes, I have 25 observations on 4 variables indeed.

`read_delim` won't work (try it and see what happens), because that would require the values to be separated by *exactly one* space.

■

(b) Run a regression to predict carbon monoxide from the other variables, and obtain a summary of the output.

Solution

The word "summary" is meant to be a big clue that `summary` is what you need:

```r
cigs.1 <- lm(co ~ tar + nicotine + weight, data = cigs)
summary(cigs.1)
```

```
Call:
lm(formula = co ~ tar + nicotine + weight, data = cigs)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.89261 -0.78269  0.00428  0.92891  2.45082 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   3.2022     3.4618   0.925 0.365464    
tar           0.9626     0.2422   3.974 0.000692 ***
nicotine     -2.6317     3.9006  -0.675 0.507234    
weight       -0.1305     3.8853  -0.034 0.973527    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.446 on 21 degrees of freedom
Multiple R-squared:  0.9186,	Adjusted R-squared:  0.907 
F-statistic: 78.98 on 3 and 21 DF,  p-value: 1.329e-11
```

■

(c) Which one of your explanatory variables would you remove from this regression? Explain (very) briefly. Go ahead and fit the regression without it, and describe how the change in R-squared from the regression in (b) was entirely predictable.

Solution

First, the x-variable to remove. The obvious candidate is `weight`, since it has easily the highest, and clearly non-significant, P-value. So, out it comes:

```r
cigs.2 <- lm(co ~ tar + nicotine, data = cigs)
summary(cigs.2)
```

```
Call:
lm(formula = co ~ tar + nicotine, data = cigs)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.89941 -0.78470 -0.00144  0.91585  2.43064 
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0896     0.8438   3.662 0.001371 **
tar           0.9625     0.2367   4.067 0.000512 ***
nicotine     -2.6463     3.7872  -0.699 0.492035
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.413 on 22 degrees of freedom
Multiple R-squared:  0.9186,    Adjusted R-squared:  0.9112
F-statistic: 124.1 on 2 and 22 DF,  p-value: 1.042e-12
```

R-squared has dropped from 0.9186 to …0.9186! That is, taking out `weight` has not just had a minimal effect on R-squared; it's not changed R-squared at all. This is because `weight` was so far from being significant: it literally had *nothing* to add.

Another way of achieving the same thing is via the function `update`, which takes a fitted model object and describes the *change* that you want to make:

```r
cigs.2a <- update(cigs.1, . ~ . - weight)
summary(cigs.2a)
```

```
Call:
lm(formula = co ~ tar + nicotine, data = cigs)

Residuals:
     Min       1Q   Median       3Q      Max
-2.89941 -0.78470 -0.00144  0.91585  2.43064

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0896     0.8438   3.662 0.001371 **
tar           0.9625     0.2367   4.067 0.000512 ***
nicotine     -2.6463     3.7872  -0.699 0.492035
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.413 on 22 degrees of freedom
Multiple R-squared:  0.9186,    Adjusted R-squared:  0.9112
F-statistic: 124.1 on 2 and 22 DF,  p-value: 1.042e-12
```

This can be shorter than describing the whole model again, as you do with the `cigs.2` version of `lm`. The syntax is that you first specify a "base" fitted model object that you're going to update. Because the model `cigs.1` contains all the information about the kind of model it is, and which data frame the data come from, R already knows that this is a linear multiple regression and which x's it contains. The second thing to describe is the change from the "base". In this case, we want to use the same response variable and all the same explanatory variables that we had before, except for `weight`. This is specified by a special kind of model formula where `.` means "whatever was there before": in English, "same response and same explanatories except take out `weight`".

■

(d) Fit a regression predicting carbon monoxide from `nicotine` *only*, and display the summary.

Solution

As you would guess:

```
cigs.3 <- lm(co ~ nicotine, data = cigs)
summary(cigs.3)
```

```
Call:
lm(formula = co ~ nicotine, data = cigs)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3273 -1.2228  0.2304  1.2700  3.9357 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.6647     0.9936   1.675    0.107    
nicotine     12.3954     1.0542  11.759 3.31e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.828 on 23 degrees of freedom
Multiple R-squared:  0.8574,	Adjusted R-squared:  0.8512 
F-statistic: 138.3 on 1 and 23 DF,  p-value: 3.312e-11
```

■

(e) `nicotine` was far from being significant in the model of (c), and yet in the model of (d), it was *strongly* significant, and the R-squared value of (d) was almost as high as that of (c). What does this say about the importance of `nicotine` as an explanatory variable? Explain, as briefly as you can manage.

Solution

What this says is that you *cannot* say anything about the "importance" of `nicotine` without also describing the context that you're talking about. *By itself*, `nicotine` is important, but *when you have* ***`tar`*** *in the model*, `nicotine` is not important: precisely, it now has nothing to add over and above the predictive value that `tar` has. You might guess that this is because `tar` and `nicotine` are "saying the same thing" in some fashion. We'll explore that in a moment.

■

(f) Make a "pairs plot": that is, scatter plots between all pairs of variables. This can be done by feeding the whole data frame into `plot`.[5] Do you see any strong relationships that do *not* include `co`? Does that shed any light on the last part? Explain briefly (or "at length" if that's how it comes out).

Solution

Plot the entire data frame:

```
plot(cigs)
```

[5]This is a base graphics graph rather than a `ggplot` one, but it will do for our purposes.

We're supposed to ignore `co`, but I comment that strong relationships between `co` and *both* of `tar` and `nicotine` show up here, along with `weight` being at most weakly related to anything else.

That leaves the relationship of `tar` and `nicotine` with each other. That also looks like a strong linear trend. When you have correlations between explanatory variables, it is called "multicollinearity".

Having correlated x's is trouble. Here is where we find out why. The problem is that when `co` is large, `nicotine` is large, and a large value of `tar` will come along with it. So we don't know whether a large value of `co` is caused by a large value of `tar` or a large value of `nicotine`: there is no way to separate out their effects because in effect they are "glued together".

You might know of this effect (in an experimental design context) as "confounding": the effect of `tar` on `co` is confounded with the effect of `nicotine` on `co`, and you can't tell which one deserves the credit for predicting `co`.

If you were able to design an experiment here, you could (in principle) manufacture a bunch of cigarettes with high tar; some of them would have high nicotine and some would have low. Likewise for low tar. Then the correlation between `nicotine` and `tar` would go away, their effects on `co` would no longer be confounded, and you could see unambiguously which one of the variables deserves credit for predicting `co`. Or maybe it depends on both, genuinely, but at least then you'd know.

We, however, have an observational study, so we have to make do with the data we have. Confounding is one of the risks we take when we work with observational data.

This was a "base graphics" plot. There is a way of doing a `ggplot`-style "pairs plot", as this is called, thus:

```
library(GGally)
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
cigs %>% ggpairs(progress = FALSE)
```

As ever, `install.packages` first, in the likely event that you don't have this package installed yet. Once you do, though, I think this is a nicer way to get a pairs plot.

This plot is a bit more sophisticated: instead of just having the scatterplots of the pairs of variables in the row and column, it uses the diagonal to show a "kernel density" (a smoothed-out histogram), and upper-right it shows the correlation between each pair of variables. The three correlations between `co`, `tar` and `nicotine` are clearly the highest.

If you want only some of the columns to appear in your pairs plot, `select` them first, and then pass that data frame into `ggpairs`. Here, we found that `weight` was not correlated with anything much, so we can take it out and then make a pairs plot of the other variables:

```
cigs %>% select(-weight) %>% ggpairs(progress = FALSE)
```

The three correlations that remain are all very high, which is entirely consistent with the strong linear relationships that you see bottom left.

■

16.17 Maximal oxygen uptake in young boys

A physiologist wanted to understand the relationship between physical characteristics of pre-adolescent boys and their maximal oxygen uptake (millilitres of oxygen per kilogram of body weight). The data are in link for a random sample of 10 pre-adolescent boys. The variables are (with units):

- `uptake`: Oxygen uptake (millitres of oxygen per kilogram of body weight)
- `age`: boy's age (years)
- `height`: boy's height (cm)
- `weight`: boy's weight (kg)
- `chest`: chest depth (cm).

(a) Read the data into R and confirm that you do indeed have 10 observations.

Solution

```r
my_url <- "http://ritsokiguess.site/datafiles/youngboys.txt"
boys <- read_delim(my_url, " ")
```

```
Rows: 10 Columns: 5
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (5): uptake, age, height, weight, chest

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
boys
```

```
# A tibble: 10 x 5
   uptake   age height weight chest
    <dbl> <dbl>  <dbl>  <dbl> <dbl>
 1   1.54   8.4   132    29.1  14.4
 2   1.74   8.7   136.   29.7  14.5
 3   1.32   8.9   128.   28.4  14
 4   1.5    9.9   131.   28.8  14.2
 5   1.46   9     130    25.9  13.6
 6   1.35   7.7   128.   27.6  13.9
 7   1.53   7.3   130.   29    14
 8   1.71   9.9   138.   33.6  14.6
 9   1.27   9.3   127.   27.7  13.9
10   1.5    8.1   132.   30.8  14.5
```

10 boys (rows) indeed.

■

(b) Fit a regression predicting oxygen uptake from all the other variables, and display the results.

Solution

Fitting four explanatory variables with only ten observations is likely to be pretty shaky, but we press ahead regardless:

```r
boys.1 <- lm(uptake ~ age + height + weight + chest, data = boys)
summary(boys.1)
```

```
Call:
lm(formula = uptake ~ age + height + weight + chest, data = boys)

Residuals:
        1         2         3         4         5         6         7         8
-0.020697  0.019741 -0.003649  0.038470 -0.023639 -0.026026  0.050459 -0.014380
        9        10
 0.004294 -0.024573

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.774739   0.862818  -5.534 0.002643 **
age         -0.035214   0.015386  -2.289 0.070769 .
height       0.051637   0.006215   8.308 0.000413 ***
weight      -0.023417   0.013428  -1.744 0.141640
chest        0.034489   0.085239   0.405 0.702490
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03721 on 5 degrees of freedom
Multiple R-squared:  0.9675,    Adjusted R-squared:  0.9415
F-statistic:  37.2 on 4 and 5 DF,  p-value: 0.0006513
```

■

(c) (A one-mark question.) Would you say, on the evidence so far, that the regression fits well or badly? Explain (very) briefly.

Solution

R-squared of 0.97 (97%) is very high, so I'd say this regression fits very well. That's all. I said "on the evidence so far" to dissuade you from overthinking this, or thinking that you needed to produce some more evidence. That, plus the fact that this was only one mark.

■

(d) It seems reasonable that an older boy should have a greater oxygen uptake, all else being equal. Is this supported by your output? Explain briefly.

Solution

If an older boy has greater oxygen uptake (the "all else equal" was a hint), the slope of `age` should be positive. It is not: it is -0.035, so it is suggesting (all else equal) that a greater age goes with a *smaller* oxygen uptake. The reason why this happens (which you didn't need,

but you can include it if you like) is that `age` has a non-small P-value of 0.07, so that the `age` slope is not significantly different from zero. With all the other variables, `age` has nothing to *add* over and above them, and we could therefore remove it.

■

(e) It seems reasonable that a boy with larger weight should have larger lungs and thus a *statistically significantly* larger oxygen uptake. Is that what happens here? Explain briefly.

Solution

Look at the P-value for `weight`. This is 0.14, not small, and so a boy with larger weight does not have a significantly larger oxygen uptake, all else equal. (The slope for `weight` is not significantly different from zero either.) I emphasized "statistically significant" to remind you that this means to do a test and get a P-value.

■

(f) Fit a model that contains only the significant explanatory variables from your first regression. How do the R-squared values from the two regressions compare? (The last sentence asks for more or less the same thing as the next part. Answer it either here or there. Either place is good.)

Solution

Only `height` is significant, so that's the only explanatory variable we need to keep. I would just do the regression straight rather than using `update` here:

```
boys.2 <- lm(uptake ~ height, data = boys)
summary(boys.2)
```

```
Call:
lm(formula = uptake ~ height, data = boys)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.069879 -0.033144  0.001407  0.009581  0.084012 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -3.843326   0.609198  -6.309 0.000231 ***
height       0.040718   0.004648   8.761 2.26e-05 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05013 on 8 degrees of freedom
Multiple R-squared:  0.9056,    Adjusted R-squared:  0.8938
F-statistic: 76.75 on 1 and 8 DF,  p-value: 2.258e-05
```

If you want, you can use `update` here, which looks like this:

```
boys.2a <- update(boys.1, . ~ . - age - weight - chest)
summary(boys.2a)
```

```
Call:
lm(formula = uptake ~ height, data = boys)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.069879 -0.033144  0.001407  0.009581  0.084012 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -3.843326   0.609198  -6.309 0.000231 ***
height       0.040718   0.004648   8.761 2.26e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05013 on 8 degrees of freedom
Multiple R-squared:  0.9056,    Adjusted R-squared:  0.8938
F-statistic: 76.75 on 1 and 8 DF,  p-value: 2.258e-05
```

This doesn't go quite so smoothly here because there are three variables being removed, and it's a bit of work to type them all.

■

(g) How has R-squared changed between your two regressions? Describe what you see in a few words.

Solution

R-squared has dropped by a bit, from 97% to 91%. (Make your own call: pull out the two R-squared numbers, and say a word or two about how they compare. I don't much mind what

you say: "R-squared has decreased (noticeably)", "R-squared has hardly changed". But say something.)

■

(h) Carry out a test comparing the fit of your two regression models. What do you conclude, and therefore what recommendation would you make about the regression that would be preferred?

Solution

The word "test" again implies something that produces a P-value with a null hypothesis that you might reject. In this case, the test that compares two models differing by more than one x uses `anova`, testing the null hypothesis that the two regressions are equally good, against the alternative that the bigger (first) one is better. Feed `anova` two fitted model objects, smaller first:

```
anova(boys.2, boys.1)
```

```
Analysis of Variance Table

Model 1: uptake ~ height
Model 2: uptake ~ age + height + weight + chest
  Res.Df       RSS Df Sum of Sq      F Pr(>F)
1      8 0.0201016
2      5 0.0069226  3  0.013179 3.1729  0.123
```

This P-value of 0.123 is not small, so we do not reject the null hypothesis. There is not a significant difference in fit between the two models. Therefore, we should go with the smaller model `boys.2` because it is simpler.

That drop in R-squared from 97% to 91% was, it turns out, *not* significant: the three extra variables could have produced a change in R-squared like that, *even if they were worthless*.[6]

If you have learned about "adjusted R-squared", you might recall that this is supposed to go down *only* if the variables you took out should not have been taken out. But adjusted R-squared goes down here as well, from 94% to 89% (not quite as much, therefore). What happens is that adjusted R-squared is rather more relaxed about keeping variables than the `anova` F-test is; if we had used an α of something like 0.10, the decision between the two models would have been a lot closer, and this is reflected in the adjusted R-squared values.

■

[6] Recall that adding x-variables to a regression will always make R-squared go up, even if they are just random noise.

(i) Obtain a table of correlations between all the variables in the data frame. Do this by feeding the whole data frame into `cor`. We found that a regression predicting oxygen uptake from just `height` was acceptably good. What does your table of correlations say about why that is? (Hint: look for all the correlations that are *large*.)

Solution

Correlations first:

```
cor(boys)
```

```
          uptake       age    height    weight     chest
uptake 1.0000000 0.1361907 0.9516347 0.6576883 0.7182659
age    0.1361907 1.0000000 0.3274830 0.2307403 0.1657523
height 0.9516347 0.3274830 1.0000000 0.7898252 0.7909452
weight 0.6576883 0.2307403 0.7898252 1.0000000 0.8809605
chest  0.7182659 0.1657523 0.7909452 0.8809605 1.0000000
```

The correlations with `age` are all on the low side, but all the other correlations are high, not just between `uptake` and the other variables, but between the explanatory variables as well.

Why is this helpful in understanding what's going on? Well, imagine a boy with large height (a tall one). The regression `boys.2` says that this alone is enough to predict that such a boy's oxygen uptake is likely to be large, since the slope is positive. But the correlations tell you more: a boy with large height is also (somewhat) likely to be older (have large age), heavier (large weight) and to have larger `chest` cavity. So oxygen uptake does depend on those other variables as well, but once you know `height` you can make a good guess at their values; you don't need to know them.

Further remarks: `age` has a low correlation with `uptake`, so its non-significance earlier appears to be "real": it really does have nothing extra to say, because the other variables have a stronger link with `uptake` than `age`. Height, however, seems to be the best way of relating oxygen uptake to any of the other variables. I think the suppositions from earlier about relating oxygen uptake to "bigness"[7] in some sense are actually sound, but age and weight and `chest` capture "bigness" worse than height does. Later, when you learn about Principal Components, you will see that the first principal component, the one that best captures how the variables vary together, is often "bigness" in some sense.

Another way to think about these things is via pairwise scatterplots. The nicest way to produce these is via `ggpairs` from package `GGally`:

[7]This is not, I don't think, a real word, but I mean size emphasizing how big a boy is generally, rather than how small.

```r
boys %>% ggpairs(progress = FALSE)
```

A final remark: with five variables, we really ought to have more than ten observations (something like 50 would be better). But with more observations and the same correlation structure, the same issues would come up again, so the question would not be materially changed.

■

16.18 Facebook friends and grey matter

Is there a relationship between the number of Facebook friends a person has, and the density of grey matter in the areas of the brain associated with social perception and associative memory? To find out, a 2012 study measured both of these variables for a sample of 40 students at City University in London (England). The data are at link. The grey matter density is on a z-score standardized scale. The values are separated by *tabs*.

The aim of this question is to produce an R Markdown report that contains your answers to the questions below.

You should aim to make your report flow smoothly, so that it would be pleasant for a grader to read, and can stand on its own as an analysis (rather than just being the answer to a question that I set you). Some suggestions: give your report a title and arrange it into sections with an Introduction; add a small amount of additional text here and there explaining what you are

doing and why. I don't expect you to spend a large amount of time on this, but I *do* hope you will make some effort. (My report came out to 4 Word pages.)

(a) Read in the data and make a scatterplot for predicting the number of Facebook friends from the grey matter density. On your scatterplot, add a smooth trend.

Solution

Begin your document with a code chunk containing `library(tidyverse)`. The data values are separated by tabs, which you will need to take into account:

```
my_url <- "http://ritsokiguess.site/datafiles/facebook.txt"
fb <- read_tsv(my_url)
```

```
Rows: 40 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: "\t"
dbl (2): GMdensity, FBfriends

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
fb
```

```
# A tibble: 40 x 2
   GMdensity FBfriends
       <dbl>     <dbl>
 1      -1.8        23
 2       0.1        35
 3      -1.2        80
 4      -0.4       110
 5      -0.9       120
 6      -2.1       140
 7      -1.5       168
 8       0.5       132
 9       0.6       154
10      -0.5       241
# i 30 more rows
```

```
ggplot(fb, aes(x = GMdensity, y = FBfriends)) + geom_point() + geom_smooth()
```

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

■

(b) Describe what you see on your scatterplot: is there a trend, and if so, what kind of trend is it? (Don't get too taken in by the exact shape of your smooth trend.) Think "form, direction, strength".

Solution

I'd say there seems to be a weak, upward, apparently linear trend. The points are not especially close to the trend, so I don't think there's any justification for calling this other than "weak". (If you think the trend is, let's say, "moderate", you ought to say what makes you think that: for example, that the people with a lot of Facebook friends also tend to have a higher grey matter density. I can live with a reasonably-justified "moderate".) The reason I said not to get taken in by the shape of the smooth trend is that this has a "wiggle" in it: it goes down again briefly in the middle. But this is likely a quirk of the data, and the trend, if there is any, seems to be an upward one.

■

(c) Fit a regression predicting the number of Facebook friends from the grey matter density, and display the output.

Solution

That looks like this. You can call the "fitted model object" whatever you like, but you'll need to get the capitalization of the variable names correct:

```
fb.1 <- lm(FBfriends ~ GMdensity, data = fb)
summary(fb.1)
```

```
Call:
lm(formula = FBfriends ~ GMdensity, data = fb)

Residuals:
    Min      1Q  Median      3Q     Max 
-339.89 -110.01   -5.12   99.80  303.64 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   366.64      26.35  13.916  < 2e-16 ***
GMdensity      82.45      27.58   2.989  0.00488 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 165.7 on 38 degrees of freedom
Multiple R-squared:  0.1904,	Adjusted R-squared:  0.1691 
F-statistic: 8.936 on 1 and 38 DF,  p-value: 0.004882
```

I observe, though I didn't ask you to, that the R-squared is pretty awful, going with a correlation of

```
sqrt(0.1904)
```

```
[1] 0.4363485
```

which *would* look like as weak of a trend as we saw.[8]

■

(d) Is the slope of your regression line significantly different from zero? What does that mean, in the context of the data?

[8]Correlations have to go up beyond 0.50 before they start looking at all interesting.

Solution

The P-value of the slope is 0.005, which is less than 0.05. Therefore the slope *is* significantly different from zero. That means that the number of Facebook friends really does depend on the grey matter density, for the whole population of interest and not just the 40 students observed here (that were a sample from that population). I don't mind so much what you think the population is, but it needs to be clear that the relationship applies to a population. Another way to approach this is to say that you would expect this relationship to show up again in another similar experiment. That also works, because it gets at the idea of reproducibility.

■

(e) Are you surprised by the results of parts (b) and (d)? Explain briefly.

Solution

I *am* surprised, because I thought the trend on the scatterplot was so weak that there would not be a significant slope. I guess there was enough of an upward trend to be significant, and with $n = 40$ observations we were able to get a significant slope out of that scatterplot. With this many observations, even a weak correlation can be significantly nonzero. You can be surprised or not, but you need to have some kind of consideration of the strength of the trend on the scatterplot as against the significance of the slope. For example, if you decided that the trend was "moderate" in strength, you would be justified in being less surprised than I was. Here, there is the usual issue that we have proved that the slope is not zero (that the relationship is not flat), but we may not have a very clear idea of what the slope actually *is*. There are a couple of ways to get a confidence interval. The obvious one is to use R as a calculator and go up and down twice its standard error (to get a rough idea):

```
82.45 + 2 * 27.58 * c(-1, 1)
```

```
[1]  27.29 137.61
```

The `c()` thing is to get both confidence limits at once. The smoother way is this:

```
confint(fb.1)
```

```
                2.5 %   97.5 %
(Intercept) 313.30872 419.9810
GMdensity    26.61391 138.2836
```

Feed `confint` a "fitted model object" and it'll give you confidence intervals (by default 95%) for all the parameters in it.

The confidence interval for the slope goes from about 27 to about 138. That is to say, a one-unit increase in grey matter density goes with an increase in Facebook friends of this much. This is not especially insightful: it's bigger than zero (the test was significant), but other than that, it could be almost anything. *This* is where the weakness of the trend comes back to bite us. With this much scatter in our data, we need a *much* larger sample size to estimate accurately how big an effect grey matter density has.

■

(f) Obtain a scatterplot with the regression line on it.

Solution

Just a modification of (a):

```
ggplot(fb, aes(x = GMdensity, y = FBfriends)) + geom_point() +
  geom_smooth(method = "lm")
```

```
`geom_smooth()` using formula = 'y ~ x'
```

■

(g) Obtain a plot of the residuals from the regression against the fitted values, and comment briefly on it.

Solution

This is, to my mind, the easiest way:

```
ggplot(fb.1, aes(x = .fitted, y = .resid)) + geom_point()
```

There is some "magic" here, since the fitted model object is not actually a data frame, but it works this way. That looks to me like a completely random scatter of points. Thus, I am completely happy with the straight-line regression that we fitted, and I see no need to improve it.

(You should make two points here: one, describe what you see, and two, what it implies about whether or not your regression is satisfactory.)

Compare that residual plot with this one:

```
ggplot(fb.1, aes(x = .fitted, y = .resid)) +
  geom_point() + geom_smooth()
```

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

Now, why did I try adding a smooth trend, and why is it not necessarily a good idea? The idea of a residual plot is that there should be no trend, and so the smooth trend curve ought to go straight across. The problem is that it will tend to wiggle, just by chance, as here: it looks as if it goes up and down before flattening out. But if you look at the points, *they* are all over the place, not close to the smooth trend at all. So the smooth trend is rather deceiving. Or, to put it another way, to indicate a real problem, the smooth trend would have to be a *lot* farther from flat than this one is. I'd call this one basically flat.

■

16.19 Endogenous nitrogen excretion in carp

A paper in Fisheries Science reported on variables that affect "endogenous nitrogen excretion" or ENE in carp raised in Japan. A number of carp were divided into groups based on body weight, and each group was placed in a different tank. The mean body weight of the carp placed in each tank was recorded. The carp were then fed a protein-free diet three times daily for a period of 20 days. At the end of the experiment, the amount of ENE in each tank was measured, in milligrams of total fish body weight per day. (Thus it should not matter that some of the tanks had more fish than others, because the scaling is done properly.)

For this question, write a report in R Markdown that answers the questions below and contains some narrative that describes your analysis. Create an HTML document from your R Markdown.

(a) Read the data in from link. There are 10 tanks.

Solution

Just this. Listing the data is up to you, but doing so and commenting that the values appear to be correct will improve your report.

```
my_url <- "http://ritsokiguess.site/datafiles/carp.txt"
carp <- read_delim(my_url, " ")
```

```
Rows: 10 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (3): tank, bodyweight, ENE

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
carp
```

```
# A tibble: 10 x 3
    tank bodyweight   ENE
   <dbl>      <dbl> <dbl>
 1     1       11.7  15.3
 2     2       25.3   9.3
 3     3       90.2   6.5
 4     4      213     6
 5     5       10.2  15.7
 6     6       17.6  10
 7     7       32.6   8.6
 8     8       81.3   6.4
 9     9      142.    5.6
10    10      286.    6
```

■

(b) Create a scatterplot of ENE (response) against bodyweight (explanatory). Add a smooth trend to your plot.

Solution

```
ggplot(carp, aes(x = bodyweight, y = ENE)) + geom_point() +
  geom_smooth()
```

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

This part is just about getting the plot. Comments are coming in a minute. Note that `ENE` is capital letters, so that `ene` will not work.

■

(c) Is there an upward or downward trend (or neither)? Is the relationship a line or a curve? Explain briefly.

Solution

The trend is downward: as bodyweight increases, ENE decreases. However, the decrease is rapid at first and then levels off, so the relationship is nonlinear. I want some kind of support for an assertion of non-linearity: anything that says that the slope or rate of decrease is not constant is good.

■

(d) Fit a straight line to the data, and obtain the R-squared for the regression.

Solution

`lm`. The first stage is to fit the straight line, saving the result in a variable, and the second stage is to look at the "fitted model object", here via `summary`:

```
carp.1 <- lm(ENE ~ bodyweight, data = carp)
summary(carp.1)
```

```
Call:
lm(formula = ENE ~ bodyweight, data = carp)

Residuals:
   Min     1Q Median     3Q    Max 
-2.800 -1.957 -1.173  1.847  4.572 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 11.40393    1.31464   8.675 2.43e-05 ***
bodyweight  -0.02710    0.01027  -2.640   0.0297 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.928 on 8 degrees of freedom
Multiple R-squared:  0.4656,	Adjusted R-squared:  0.3988 
F-statistic: 6.971 on 1 and 8 DF,  p-value: 0.0297
```

Finally, you need to give me a (suitably rounded) value for R-squared: 46.6% or 47% or the equivalents as a decimal. I just need the value at this point. This kind of R-squared is actually pretty good for natural data, but the issue is whether we can improve it by fitting a non-linear model.[9]

■

(e) Obtain a residual plot (residuals against fitted values) for this regression. Do you see any problems? If so, what does that tell you about the relationship in the data?

Solution

This is the easiest way: feed the output of the regression straight into `ggplot`:

[9]The suspicion being that we can, since the scatterplot suggested serious non-linearity.

```r
ggplot(carp.1, aes(x = .fitted, y = .resid)) + geom_point()
```

■

(f) Fit a parabola to the data (that is, including an x-squared term). Compare the R-squared values for the models in this part and part (d). Does that suggest that the parabola model is an improvement here over the linear model?

Solution

Add bodyweight-squared to the regression. Don't forget the `I()`:

```r
carp.2 <- lm(ENE ~ bodyweight + I(bodyweight^2), data = carp)
summary(carp.2)
```

```
Call:
lm(formula = ENE ~ bodyweight + I(bodyweight^2), data = carp)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.0834 -1.7388 -0.5464  1.3841  2.9976 
```

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     13.7127373  1.3062494  10.498 1.55e-05 ***
bodyweight      -0.1018390  0.0288109  -3.535  0.00954 **
I(bodyweight^2)  0.0002735  0.0001016   2.692  0.03101 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.194 on 7 degrees of freedom
Multiple R-squared:  0.7374,    Adjusted R-squared:  0.6624
F-statistic: 9.829 on 2 and 7 DF,  p-value: 0.009277
```

R-squared has gone up from 47% to 74%, a substantial improvement. This suggests to me that the parabola model is a substantial improvement.[10]

I try to avoid using the word "significant" in this context, since we haven't actually done a test of significance.

The reason for the `I()` is that the up-arrow has a special meaning in `lm`, relating to interactions between factors (as in ANOVA), that we don't want here. Putting `I()` around it means "use as is", that is, raise bodyweight to power 2, rather than using the special meaning of the up-arrow in `lm`.

Because it's the up-arrow that is the problem, this applies whenever you're raising an explanatory variable to a power (or taking a reciprocal or a square root, say).

■

(g) Is the test for the slope coefficient for the squared term significant? What does this mean?

Solution

Look along the `bodyweight`-squared line to get a P-value of 0.031. This is less than the default 0.05, so it *is* significant. This means, in short, that the quadratic model is a significant *improvement* over the linear one.[11] Said longer: the null hypothesis being tested is that the slope coefficient of the squared term is zero (that is, that the squared term has nothing to add over the linear model). This is rejected, so the squared term has *something* to add in terms of quality of prediction.

■

(h) Make the scatterplot of part (b), but add the fitted curve. Describe any way in which the curve fails to fit well.

[10] Again, not a surprise, given our initial scatterplot.

[11] Now we can use that word *significant*.

Solution

This is a bit slippery, because the points to plot and the fitted curve are from different data frames. What you do in this case is to put a `data=` in one of the `geom`s, which says "don't use the data frame that was in the `ggplot`, but use this one instead". I would think about starting with the regression object `carp.2` as my base data frame, since we want (or I want) to do two things with that: plot the fitted values and join them with lines. Then I want to add the original data, just the points:

```
ggplot(carp.2, aes(x = carp$bodyweight, y = .fitted), colour = "blue") +
  geom_line(colour = "blue") +
  geom_point(data = carp, aes(x = bodyweight, y = ENE))
```

This works, but is not very aesthetic, because the bodyweight that is plotted against the fitted values is in the wrong data frame, and so we have to use the dollar-sign thing to get it from the right one.

A better way around this is "augment" the data with output from the regression object. This is done using `augment` from package `broom`:

```
library(broom)
carp.2a <- augment(carp.2, carp)
carp.2a
```

```
# A tibble: 10 x 9
    tank bodyweight   ENE .fitted .resid  .hat .sigma .cooksd .std.resid
   <dbl>      <dbl> <dbl>   <dbl>  <dbl> <dbl>  <dbl>   <dbl>      <dbl>
 1     1       11.7  15.3   12.6   2.74  0.239   1.99 0.215        1.43
 2     2       25.3   9.3   11.3  -2.01  0.163   2.19 0.0651      -1.00
 3     3       90.2   6.5    6.75 -0.252 0.240   2.37 0.00182     -0.132
 4     4      213     6      4.43  1.57  0.325   2.24 0.122        0.871
 5     5       10.2  15.7   12.7   3.00  0.251   1.90 0.279        1.58
 6     6       17.6  10     12.0  -2.01  0.199   2.19 0.0866      -1.02
 7     7       32.6   8.6   10.7  -2.08  0.143   2.18 0.0583      -1.03
 8     8       81.3   6.4    7.24 -0.841 0.211   2.34 0.0166      -0.431
 9     9      142.    5.6    4.78  0.822 0.355   2.33 0.0398       0.466
10    10      286.    6      6.94 -0.940 0.875   2.11 3.40        -1.21
```

so now you see what `carp.2a` has in it, and then:

```
g <- ggplot(carp.2a, aes(x = bodyweight, y = .fitted)) +
  geom_line(colour = "blue") +
  geom_point(aes(y = ENE))
```

This is easier coding: there are only two non-standard things. The first is that the fitted-value lines should be a distinct colour like blue so that you can tell them from the data points. The second thing is that for the second `geom_point`, the one that plots the data, the x coordinate `bodyweight` is correct so that we don't have to change that; we only have to change the y-coordinate, which is `ENE`. The plot is this:

```
g
```

Concerning interpretation, you have a number of possibilities here. The simplest is that the points in the middle are above the curve, and the points at the ends are below. (That is, negative residuals at the ends, and positive ones in the middle, which gives you a hint for the next part.) Another is that the parabola curve fails to capture the *shape* of the relationship; for example, I see nothing much in the data suggesting that the relationship should go back up, and even given that, the fitted curve doesn't go especially near any of the points.

I was thinking that the data should be fit better by something like the left half of an upward-opening parabola, but I guess the curvature on the left half of the plot suggests that it needs most of the left half of the parabola just to cover the left half of the plot.

The moral of the story, as we see in the next part, is that the parabola is the wrong curve for the job.

■

(i) Obtain a residual plot for the parabola model. Do you see any problems with it? (If you do, I'm not asking you to do anything about them in this question, but I will.)

■

The same idea as before for the other residual plot. Use the fitted model object `carp.2` as your data frame for the `ggplot`:

```
ggplot(carp.2, aes(x = .fitted, y = .resid)) + geom_point()
```

I think this is *still* a curve (or, it goes down and then sharply up at the end). Either way, there is still a pattern.

That was all I needed, but as to what this means: our parabola was a curve all right, but it appears not to be the right *kind* of curve. I think the original data looks more like a hyperbola (a curve like $y = 1/x$) than a parabola, in that it seems to decrease fast and then gradually to a limit, and *that* suggests, as in the class example, that we should try an asymptote model. Note how I specify it, with the `I()` thing again, since `/` has a special meaning to `lm` in the same way that `^` does:

```
carp.3 <- lm(ENE ~ I(1 / bodyweight), data = carp)
summary(carp.3)
```

```
Call:
lm(formula = ENE ~ I(1/bodyweight), data = carp)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.29801 -0.12830  0.04029  0.26702  0.91707 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)       5.1804     0.2823   18.35 8.01e-08 ***
I(1/bodyweight) 107.6690     5.8860   18.29 8.21e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6121 on 8 degrees of freedom
Multiple R-squared:  0.9766,    Adjusted R-squared:  0.9737
F-statistic: 334.6 on 1 and 8 DF,  p-value: 8.205e-08
```

That fits *extraordinarily* well, with an R-squared up near 98%. The intercept is the asymptote, which suggests a (lower) limit of about 5.2 for `ENE` (in the limit for large bodyweight). We would have to ask the fisheries scientist whether this kind of thing is a reasonable biological mechanism. It says that a carp always has some ENE, no matter how big it gets, but a smaller carp will have a lot more.

Does the fitted value plot look reasonable now? This is `augment` again since the fitted values and observed data come from different data frames:

```r
library(broom)
augment(carp.3, carp) %>%
  ggplot(aes(x = bodyweight, y = .fitted)) +
  geom_line(colour = "blue") +
  geom_point(aes(y = ENE))
```

I'd say that does a really nice job of fitting the data. But it would be nice to have a few more tanks with large-bodyweight fish, to convince us that we have the shape of the trend right.

And, as ever, the residual plot. That's a lot easier than the plot we just did:

```
ggplot(carp.3, aes(x = .fitted, y = .resid)) + geom_point()
```

All in all, that looks pretty good (and certainly a vast improvement over the ones you got before).

When you write up your report, you can make it flow better by writing it in a way that suggests that each thing was the obvious thing to do next: that is, that *you* would have thought to do it next, rather than me telling you what to do.

My report (as an R Markdown file) is at link. Download it, knit it, play with it.

■

16.20 Salaries of social workers

Another salary-prediction question: does the number of years of work experience that a social worker has help to predict their salary? Data for 50 social workers are in link.

(a) Read the data into R. Check that you have 50 observations on two variables. Also do something to check that the years of experience and annual salary figures look reasonable overall.

Solution

```
my_url <- "http://ritsokiguess.site/datafiles/socwork.txt"
soc <- read_delim(my_url, " ")
```

```
Rows: 50 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): experience, salary

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
soc
```

```
# A tibble: 50 x 2
   experience salary
        <dbl>  <dbl>
 1          7  26075
 2         28  79370
 3         23  65726
 4         18  41983
 5         19  62308
 6         15  41154
 7         24  53610
 8         13  33697
 9          2  22444
10          8  32562
# i 40 more rows
```

That checks that we have the right *number* of observations; to check that we have sensible *values*, something like `summary` is called for:

```
summary(soc)
```

```
   experience        salary
 Min.   : 1.00   Min.   :16105
 1st Qu.:13.50   1st Qu.:36990
 Median :20.00   Median :50948
 Mean   :18.12   Mean   :50171
 3rd Qu.:24.75   3rd Qu.:65204
 Max.   :28.00   Max.   :99139
```

A person working in any field cannot have a negative number of years of experience, and cannot have more than about 40 years of experience (or else they would have retired). Our experience numbers fit that. Salaries had better be five or six figures, and salaries for social workers are not generally all that high, so these figures look reasonable.

A rather more `tidyverse` way is this:

```
soc %>%
  summarize(across(everything(),
                   list(min = \(x) min(x),  max = \(x) max(x))))
```

```
# A tibble: 1 x 4
  experience_min experience_max salary_min salary_max
           <dbl>          <dbl>      <dbl>      <dbl>
1              1             28      16105      99139
```

This gets the minimum and maximum of all the variables. I would have liked them arranged in a nice rectangle (`min` and `max` as rows, the variables as columns), but that's not how this came out. We fix that shortly.

The code so far uses `across`. This means to do something across multiple columns. In this case, we want to do the calculation on *all* the columns, so we use the select-helper `everything`. You can use any of the other select-helpers like `starts_with`, or you could do something like `where(is.numeric)` to do your summaries only on the quantitative columns (which would also work here). The thing after the `everything()` means "for each column selected, work out the `min` and `max` of it"; `x` is our name for "the variable we are looking at at the moment".

What, you want a nice rectangle? This is a pivot-longer, but a fancy version because the column names encode two kinds of things, a variable and a statistic. I took the view that I wanted variables in columns (as usual), and the different summary statistics in rows. This means that the first part of the column names we created above (eg. the `salary` part of `salary_min`) should stay in columns, and the rest of it should be pivoted longer. That means using the special name `.value` for the things that should stay as columns:

```r
soc %>%
  summarize(across(everything(),
                   list(min = \(x) min(x),  max = \(x) max(x)))) %>%
  pivot_longer(everything(),
               names_to = c(".value", "statistic"),
               names_sep = "_"
               )
```

```
# A tibble: 2 x 3
  statistic experience salary
  <chr>          <dbl>  <dbl>
1 min                1  16105
2 max               28  99139
```

Note that we're using two simpler tools here, rather than one complicated one: first we get the summary statistics, and once we have that, we can do some tidying to get it arranged the way we want.

Your first guess is likely to be to make it *too* long:

```r
soc %>%
  summarize(across(everything(),
                   list(min = \(x) min(x),  max = \(x) max(x)))) %>%
  pivot_longer(everything(),
               names_to = c("variable", "statistic"),
               names_sep = "_",
               values_to = "value"
               )
```

```
# A tibble: 4 x 3
  variable   statistic value
  <chr>      <chr>     <dbl>
1 experience min           1
2 experience max          28
3 salary     min       16105
4 salary     max       99139
```

and then you'll have to make it wider, or recall that you can do the thing with `.value`. We are working "columnwise", doing something for each column, no matter how many there are. My go-to for this stuff is here.

Another way to work is with the five-number summary. This gives a more nuanced picture of the data values we have.[12]

The base-R five-number summary looks like this:

```
qq <- quantile(soc$experience)
qq
```

```
   0%   25%   50%   75%  100% 
 1.00 13.50 20.00 24.75 28.00 
```

This is what's known as a "named vector". The numbers on the bottom are the summaries themselves, and the names above say which percentile you are looking at. Unfortunately, the `tidyverse` doesn't like names, so modelling after the above doesn't quite work:

```
soc %>%
  summarize(across(everything(), list(q = \(x) quantile(x))))
```

```
Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in 
dplyr 1.1.0.
i Please use `reframe()` instead.
i When switching from `summarise()` to `reframe()`, remember that `reframe()`
  always returns an ungrouped data frame and adjust accordingly.

# A tibble: 5 x 2
  experience_q salary_q
         <dbl>    <dbl>
1          1     16105
2         13.5   36990.
3         20     50948.
4         24.8   65204.
5         28     99139
```

You can guess which percentile is which (they have to be in order), but this is not completely satisfactory. It also gives a warning because the summary is five numbers long, rather than only one (like the mean, for example), and this is not the preferred way to handle this.

The warning mentions `reframe`, which is new (as in, less than a year old as I write this). Let's see how it goes here:

[12] This might be overkill at this point, since we really only care about whether our data values are reasonable, and often just looking at the highest and lowest values will tell us that.

```r
soc %>%
  reframe(q_exp = quantile(experience), q_sal = quantile(salary))
```

```
# A tibble: 5 x 2
  q_exp  q_sal
  <dbl>  <dbl>
1   1   16105
2  13.5 36990.
3  20   50948.
4  24.8 65204.
5  28   99139
```

The idea is that `reframe` is like `summarize`, but it is designed for when your summary function returns *more than one* number, not just one number per group like `mean` or `median` do.

This is not quite the best (I don't see the percentiles and I have to repeat myself), but at least I no longer get a warning. Here's how you do it with `across`:

```r
soc %>%
  reframe(across(everything(), \(x) enframe(quantile(x)), .unpack = TRUE))
```

```
# A tibble: 5 x 4
  experience_name experience_value salary_name salary_value
  <chr>                      <dbl> <chr>              <dbl>
1 0%                           1   0%                16105
2 25%                         13.5 25%               36990.
3 50%                         20   50%               50948.
4 75%                         24.8 75%               65204.
5 100%                        28   100%              99139
```

The `enframe` turns a "named vector" (that is, a thing like my `qq` above) into a dataframe with two columns, one called `name` with the names (percentiles), and one called `value` with the values. By using `across`, you get those two columns for *each* variable, and you can see which of the five numbers is which percentile in each case.

■

(b) Make a scatterplot showing how salary depends on experience. Does the nature of the trend make sense?

Solution

The usual:

```r
ggplot(soc, aes(x = experience, y = salary)) + geom_point()
```

As experience goes up, salary also goes up, as you would expect. Also, the trend seems more or less straight.

■

(c) Fit a regression predicting salary from experience, and display the results. Is the slope positive or negative? Does that make sense?

Solution

```r
soc.1 <- lm(salary ~ experience, data = soc)
summary(soc.1)
```

```
Call:
lm(formula = salary ~ experience, data = soc)

Residuals:
     Min       1Q   Median       3Q      Max 
-17666.3  -5498.2   -726.7   4667.7  27811.6 
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11368.7     3160.3   3.597 0.000758 ***
experience    2141.4      160.8  13.314  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8642 on 48 degrees of freedom
Multiple R-squared:  0.7869,    Adjusted R-squared:  0.7825
F-statistic: 177.3 on 1 and 48 DF,  p-value: < 2.2e-16
```

The slope is (significantly) positive, which squares with our guess (more experience goes with greater salary), and also the upward trend on the scatterplot. The value of the slope is about 2,000; this means that one more year of experience goes with about a $2,000 increase in salary.

■

(d) Obtain and plot the residuals against the fitted values. What problem do you see?

Solution

The easiest way to do this with `ggplot` is to plot the *regression object* (even though it is not actually a data frame), and plot the `.fitted` and `.resid` columns in it, not forgetting the initial dots:

```
ggplot(soc.1, aes(x = .fitted, y = .resid)) + geom_point()
```

I see a "fanning-out": the residuals are getting bigger *in size* (further away from zero) as the fitted values get bigger. That is, when the (estimated) salary gets larger, it also gets more variable.

Fanning-out is sometimes hard to see. What you can do if you suspect that it might have happened is to plot the *absolute value* of the residuals against the fitted values. The absolute value is the residual without its plus or minus sign, so if the residuals are getting bigger in size, their absolute values are getting bigger. That would look like this:

```
ggplot(soc.1, aes(x = .fitted, y = abs(.resid))) + geom_point() + geom_smooth()
```

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

I added a smooth trend to this to help us judge whether the absolute-value-residuals are getting bigger as the fitted values get bigger. It looks to me as if the overall trend is an increasing one, apart from those few small fitted values that have larger-sized residuals. Don't get thrown off by the kinks in the smooth trend. Here is a smoother version:

```
ggplot(soc.1, aes(x = .fitted, y = abs(.resid))) + geom_point() + geom_smooth(span = 2)
```

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

The larger fitted values, according to this, have residuals larger in size.

The thing that controls the smoothness of the smooth trend is the value of `span` in `geom_smooth`. The default is 0.75. The larger the value you use, the smoother the trend; the smaller, the more wiggly. I'm inclined to think that the default value is a bit too small. Possibly this value is too big, but it shows you the idea.

■

(e) The problem you unearthed in the previous part is often helped by a transformation. Run Box-Cox on your data to find a suitable transformation. What transformation is suggested?

Solution

You'll need to load (and install if necessary) the package `MASS` that contains `boxcox`:

```
library(MASS)
```

When you run this, you may see a warning containing the word "masked". I talk about that below.

```
boxcox(salary ~ experience, data = soc)
```

That one looks like $\lambda = 0$ or log. You could probably also justify fourth root (power 0.25), but log is a very common transformation, which people won't need much persuasion to accept.

Extra: There's one annoyance with `MASS`: it has a `select` (which I have never used), and if you load `tidyverse` first and `MASS` second, as I have done here, when you mean to run the column-selection `select`, it will actually run the `select` that comes from `MASS`, and give you an error that you will have a terrible time debugging. That's what that "masked" message was when you loaded `MASS`. This is a great place to learn about the `conflicted` package. See here for how it works. (Scroll down to under the list of files.)

If you want to insist on something like "the `select` that lives in `dplyr`", you can do that by saying `dplyr::select`. But this is kind of cumbersome if you don't need to do it.

■

(f) Use your transformed response in a regression, showing the summary.

Solution

You can do the transformation right in the `lm`, as I do below, or if you prefer, you can create a new column that is the log-salary and then use that in the `lm`. Either way is good:

```
soc.3 <- lm(log(salary) ~ experience, data = soc)
summary(soc.3)
```

```
Call:
lm(formula = log(salary) ~ experience, data = soc)

Residuals:
```

```
     Min       1Q   Median       3Q      Max 
-0.35435 -0.09046 -0.01725  0.09739  0.26355 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 9.841315   0.056356  174.63   <2e-16 ***
experience  0.049979   0.002868   17.43   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1541 on 48 degrees of freedom
Multiple R-squared:  0.8635,	Adjusted R-squared:  0.8607 
F-statistic: 303.7 on 1 and 48 DF,  p-value: < 2.2e-16
```

■

(g) Obtain and plot the residuals against the fitted values for this regression. Do you seem to have solved the problem with the previous residual plot?

Solution

As we did before, treating the regression object as if it were a data frame:

```
ggplot(soc.3, aes(x = .fitted, y = .resid)) + geom_point()
```

That, to my mind, is a horizontal band of points, so I would say yes, I have solved the fanning out.

One concern I have about the residuals is that there seem to be a couple of very negative values: that is, are the residuals normally distributed as they should be? Well, that's easy enough to check:

```
ggplot(soc.3, aes(sample = .resid)) + stat_qq() + stat_qq_line()
```

The issues here are that those bottom two values are a bit too low, and the top few values are a bit bunched up (that curve at the top). It is really not bad, though, so I am making the call that I don't think I needed to worry. Note that the transformation we found here is the same as the log-salary used by the management consultants in the backward-elimination question, and with the same effect: an extra year of experience goes with a *percent* increase in salary.

What increase? Well, the slope is about 0.05, so adding a year of experience is predicted to increase log-salary by 0.05, or to multiply actual salary by

```
exp(0.05)
```

```
[1] 1.051271
```

or to increase salary by about 5%.[13]

■

16.21 Predicting volume of wood in pine trees

In forestry, the financial value of a tree is the volume of wood that it contains. This is difficult to estimate while the tree is still standing, but the diameter is easy to measure with a tape measure (to measure the circumference) and a calculation involving π, assuming that the cross-section of the tree is at least approximately circular. The standard measurement is "diameter at breast height" (that is, at the height of a human breast or chest), defined as being 4.5 feet above the ground.

Several pine trees had their diameter measured shortly before being cut down, and for each tree, the volume of wood was recorded. The data are in link. The diameter is in inches and the volume is in cubic inches. Is it possible to predict the volume of wood from the diameter?

(a) Read the data into R and display the values (there are not very many).

Solution

Observe that the data values are separated by spaces, and therefore that `read_delim` will do it:

```
my_url <- "http://ritsokiguess.site/datafiles/pinetrees.txt"
trees <- read_delim(my_url, " ")
```

```
Rows: 10 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): diameter, volume

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
trees
```

[13] Mathematically, e^x is approximately $1 + x$ for small x, which winds up meaning that the slope in a model like this, if it is small, indicates about the percent increase in the response associated with a 1-unit change in the explanatory variable. Note that this only works with e^x and natural logs, not base 10 logs or anything like that.

```
# A tibble: 10 x 2
   diameter volume
      <dbl>  <dbl>
 1       32    185
 2       29    109
 3       24     95
 4       45    300
 5       20     30
 6       30    125
 7       26     55
 8       40    246
 9       24     60
10       18     15
```

That looks like the data file.

■

(b) Make a suitable plot.

Solution

No clues this time. You need to recognize that you have two quantitative variables, so that a scatterplot is called for. Also, the volume is the response, so that should go on the y-axis:

```
ggplot(trees, aes(x = diameter, y = volume)) + geom_point()
```

You can put a smooth trend on it if you like, which would look like this:

```
ggplot(trees, aes(x = diameter, y = volume)) +
  geom_point() + geom_smooth()
```

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

I'll take either of those for this part, though I think the smooth trend actually obscures the issue here (because there is not so much data).

■

(c) Describe what you learn from your plot about the relationship between diameter and volume, if anything.

Solution

The word "relationship" offers a clue that a scatterplot would have been a good idea, if you hadn't realized by now. I am guided by "form, direction, strength" in looking at a scatterplot:

- Form: it is an apparently linear relationship.
- Direction: it is an upward trend: that is, a tree with a larger diameter also has a larger volume of wood. (This is not very surprising.)
- Strength: I'd call this a strong (or moderate-to-strong) relationship. (We'll see in a minute what the R-squared is.)

You don't need to be as formal as this, but you *do* need to get at the idea that it is an upward trend, apparently linear, and at least fairly strong.[14]

■

[14]When this was graded, it was 3 marks, to clue you in that there are three things to say.

(d) Fit a (linear) regression, predicting volume from diameter, and obtain the `summary`. How would you describe the R-squared?

Solution

My naming convention is (usually) to call the fitted model object by the name of the response variable and a number.[15]

```
volume.1 <- lm(volume ~ diameter, data = trees)
summary(volume.1)
```

```
Call:
lm(formula = volume ~ diameter, data = trees)

Residuals:
    Min      1Q  Median      3Q     Max 
-36.497  -9.982   1.751   8.959  28.139 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -191.749     23.954  -8.005 4.35e-05 ***
diameter      10.894      0.801  13.600 8.22e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.38 on 8 degrees of freedom
Multiple R-squared:  0.9585,	Adjusted R-squared:  0.9534 
F-statistic:   185 on 1 and 8 DF,  p-value: 8.217e-07
```

R-squared is nearly 96%, so the relationship is definitely a strong one.

I also wanted to mention the `broom` package, which was installed with the `tidyverse` but which you need to load separately. It provides two handy ways to summarize a fitted model (regression, analysis of variance or whatever):

```
library(broom)
glance(volume.1)
```

```
# A tibble: 1 x 12
```

[15] I have always used dots, but in the spirit of the `tidyverse` I suppose I should use underscores.

```
  r.squared adj.r.squared sigma statistic     p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>       <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.959         0.953  20.4      185. 0.000000822     1  -43.2  92.4  93.4
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

This gives a one-line summary of a model, including things like R-squared. This is handy if you're fitting more than one model, because you can collect the one-line summaries together into a data frame and eyeball them.

The other summary is this one:

```
tidy(volume.1)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic     p.value
  <chr>          <dbl>     <dbl>     <dbl>       <dbl>
1 (Intercept)   -192.     24.0       -8.01 0.0000435
2 diameter        10.9     0.801     13.6  0.000000822
```

This gives a table of intercepts, slopes and their P-values, but the value to this one is that it is a *data frame*, so if you want to pull anything out of it, you know how to do that:[16]

```
tidy(volume.1) %>% filter(term == "diameter")
```

```
# A tibble: 1 x 5
  term     estimate std.error statistic     p.value
  <chr>       <dbl>     <dbl>     <dbl>       <dbl>
1 diameter     10.9     0.801      13.6 0.000000822
```

This gets the estimated slope and its P-value, without worrying about the corresponding things for the intercept, which are usually of less interest anyway.

■

(e) Draw a graph that will help you decide whether you trust the linearity of this regression. What do you conclude? Explain briefly.

Solution

The thing I'm fishing for is a residual plot (of the residuals against the fitted values), and on it you are looking for a random mess of nothingness:

[16]The `summary` output is more designed for looking at than for extracting things from.

```r
ggplot(volume.1, aes(x = .fitted, y = .resid)) + geom_point()
```

Make a call. You could say that there's no discernible pattern, especially with such a small data set, and therefore that the regression is fine. Or you could say that there is fanning-in: the two points on the right have residuals close to 0 while the points on the left have residuals larger in size. Say something.

I don't think you can justify a curve or a trend, because the residuals on the left are both positive and negative.

My feeling is that the residuals on the right are close to 0 because these points have noticeably larger diameter than the others, and they are *influential* points in the regression that will pull the line closer to themselves. This is why their residuals are close to zero. But I am happy with either of the points made in the paragraph under the plot.

■

(f) What would you guess would be the volume of a tree of diameter zero? Is that what the regression predicts? Explain briefly.

Solution

Logically, a tree that has diameter zero is a non-existent tree, so its volume should be zero as well. In the regression, the quantity that says what volume is when diameter is zero is the *intercept*. Here the intercept is -192, which is definitely not zero. In fact, if you look at the

P-value, the intercept is significantly *less* than zero. Thus, the model makes no logical sense for trees of small diameter. The smallest tree in the data set has diameter 18, which is not really small, I suppose, but it is a little disconcerting to have a model that makes no logical sense.

■

(g) A simple way of modelling a tree's shape is to pretend it is a cone, like this, but probably taller and skinnier:

with its base on the ground. What is the relationship between the *diameter* (at the base) and volume of a cone? (If you don't remember, look it up. You'll probably get a formula in terms of the radius, which you'll have to convert. Cite the website you used.)

Solution

According to link, the volume of a cone is $V = \pi r^2 h/3$, where V is the volume, r is the radius (at the bottom of the cone) and h is the height. The diameter is twice the radius, so replace r by $d/2$, d being the diameter. A little algebra gives

$$V = \pi d^2 h/12.$$

■

(h) Fit a regression model that predicts volume from diameter according to the formula you obtained in the previous part. You can assume that the trees in this data set are of similar heights, so that the height can be treated as a constant.
Display the results.

Solution

According to my formula, the volume depends on the diameter squared, which I include in the model thus:

```
volume.2 <- lm(volume ~ I(diameter^2), data = trees)
summary(volume.2)
```

```
Call:
lm(formula = volume ~ I(diameter^2), data = trees)

Residuals:
    Min      1Q  Median      3Q     Max 
-29.708  -9.065  -5.722   3.032  40.816 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   -30.82634   13.82243   -2.23   0.0563 .  
I(diameter^2)   0.17091    0.01342   12.74 1.36e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.7 on 8 degrees of freedom
Multiple R-squared:  0.953, Adjusted R-squared:  0.9471 
F-statistic: 162.2 on 1 and 8 DF,  p-value: 1.359e-06
```

This adds an intercept as well, which is fine (there are technical difficulties around removing the intercept).

That's as far as I wanted you to go, but (of course) I have a few comments.

The intercept here is still negative, but not significantly different from zero, which is a step forward. The R-squared for this regression is very similar to that from our linear model (the one for which the intercept made no sense). So, from that point of view, either model predicts the data well. I should look at the residuals from this one:

```
ggplot(volume.2, aes(x = .fitted, y = .resid)) + geom_point()
```

I really don't think there are any problems there.

Now, I said to assume that the trees are all of similar height. This seems entirely questionable, since the trees vary quite a bit in diameter, and you would guess that trees with bigger diameter would also be taller. It seems more plausible that the same kind of trees (pine trees in this case) would have the same "shape", so that if you knew the diameter you could *predict* the height, with larger-diameter trees being taller. Except that we don't have the heights here, so we can't build a model for that.

So I went looking in the literature. I found this paper: link. This gives several models for relationships between volume, diameter and height. In the formulas below, there is an implied "plus error" on the right, and the α_i are parameters to be estimated.

For predicting height from diameter (equation 1 in paper):

$$h = \exp(\alpha_1 + \alpha_2 d^{\alpha_3})$$

For predicting volume from height and diameter (equation 6):

$$V = \alpha_1 d^{\alpha_2} h^{\alpha_3}$$

This is a take-off on our assumption that the trees were cone-shaped, with cone-shaped trees having $\alpha_1 = \pi/12$, $\alpha_2 = 2$ and $\alpha_3 = 1$. The paper uses different units, so α_1 is not comparable,

but α_2 and α_3 are (as estimated from the data in the paper, which were for longleaf pine) quite close to 2 and 1.

Last, the actual relationship that helps us: predicting volume from just diameter (equation 5):

$$V = \alpha_1 d^{\alpha_2}$$

This is a power law type of relationship. For example, if you were willing to pretend that a tree was a cone with height proportional to diameter (one way of getting at the idea of a bigger tree typically being taller, instead of assuming constant height as we did), that would imply $\alpha_2 = 3$ here.

This is non-linear as it stands, but we can bash it into shape by taking logs:

$$\ln V = \ln \alpha_1 + \alpha_2 \ln d$$

so that log-volume has a linear relationship with log-diameter and we can go ahead and estimate it:

```r
volume.3 <- lm(log(volume) ~ log(diameter), data = trees)
summary(volume.3)
```

```
Call:
lm(formula = log(volume) ~ log(diameter), data = trees)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.40989 -0.22341  0.01504  0.10459  0.53596 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    -5.9243     1.1759  -5.038    0.001 ** 
log(diameter)   3.1284     0.3527   8.870 2.06e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3027 on 8 degrees of freedom
Multiple R-squared:  0.9077,	Adjusted R-squared:  0.8962 
F-statistic: 78.68 on 1 and 8 DF,  p-value: 2.061e-05
```

The parameter that I called α_2 above is the slope of this model, 3.13. This is a bit different from the figure in the paper, which was 2.19. I think these are comparable even though the other parameter is not (again, measurements in different units, plus, this time we need to take the log of it). I think the "slopes" are comparable because we haven't estimated our slope all that accurately:

```
confint(volume.3)
```

```
                  2.5 %    97.5 %
(Intercept)   -8.635791 -3.212752
log(diameter)  2.315115  3.941665
```

From 2.3 to 3.9. It is definitely not zero, but we are rather less sure about what it is, and 2.19 is not completely implausible.

The R-squared here, though it is less than the other ones we got, is still high. The residuals are these:

```
ggplot(volume.3, aes(x = .fitted, y = .resid)) + geom_point()
```

which again seem to show no problems. The residuals are smaller in size now because of the log transformation: the actual and predicted log-volumes are smaller numbers than the actual and predicted volumes, so the residuals are now closer to zero.

Does this model behave itself at zero? Well, roughly at least: if the diameter is very small, its log is very negative, and the predicted log-volume is also very negative (the slope is positive). So the predicted actual volume will be close to zero. If you want to make that mathematically rigorous, you can take limits, but that's the intuition. We can also do some predictions: set up a data frame that has a column called `diameter` with some diameters to predict for:

```
d <- tibble(diameter = c(1, 2, seq(5, 50, 5)))
d
```

```
# A tibble: 12 x 1
   diameter
      <dbl>
 1        1
 2        2
 3        5
 4       10
 5       15
 6       20
 7       25
 8       30
 9       35
10       40
11       45
12       50
```

and then feed that into `predictions`from package `marginaleffects`:

```
p <- cbind(predictions(volume.3, newdata = d))
p %>% select(diameter, estimate, conf.low, conf.high) -> pp
pp
```

```
   diameter   estimate   conf.low  conf.high
1         1 -5.9242712 -8.2288987 -3.6196437
2         2 -3.7558365 -5.5833436 -1.9283295
3         5 -0.8893218 -2.0885682  0.3099245
4        10  1.2791128  0.5492341  2.0089915
5        15  2.5475658  2.0829205  3.0122111
6        20  3.4475475  3.1536433  3.7414517
7        25  4.1456275  3.9446714  4.3465837
8        30  4.7160005  4.5207483  4.9112526
9        35  5.1982439  4.9512720  5.4452158
```

```
10        40  5.6159822  5.3010818  5.9308825
11        45  5.9844534  5.6010851  6.3678218
12        50  6.3140622  5.8657639  6.7623605
```

These are predicted log-volumes, so we'd better anti-log them. `log` in R is natural logs, so this is inverted using `exp`. The ends of the confidence intervals can be exp-ed as well, which I do all at once:

```
pp %>% mutate(across(-diameter, \(x) exp(x)))
```

```
   diameter     estimate     conf.low    conf.high
1         1 2.673756e-03 2.668300e-04   0.02679222
2         2 2.338088e-02 3.759973e-03   0.14539088
3         5 4.109343e-01 1.238644e-01   1.36332224
4        10 3.593450e+00 1.731926e+00   7.45579458
5        15 1.277597e+01 8.027880e+00  20.33230738
6        20 3.142323e+01 2.342124e+01  42.15914904
7        25 6.315724e+01 5.165936e+01  77.21422468
8        30 1.117205e+02 9.190435e+01 135.80942573
9        35 1.809542e+02 1.413546e+02 231.64727491
10       40 2.747831e+02 2.005537e+02 376.48662098
11       45 3.972054e+02 2.707200e+02 582.78700332
12       50 5.522839e+02 3.527515e+02 864.68090044
```

For a diameter near zero, the predicted volume appears to be near zero as well. If you don't like the scientific notation:

```
pp %>% mutate(across(-diameter, \(x) exp(x))) %>%
  mutate(across(-diameter, \(x) format(x, scientific = FALSE)))
```

```
   diameter      estimate      conf.low    conf.high
1         1   0.002673756   0.000266830   0.02679222
2         2   0.023380883   0.003759973   0.14539088
3         5   0.410934340   0.123864356   1.36332224
4        10   3.593450319   1.731926095   7.45579458
5        15  12.775966620   8.027879965  20.33230738
6        20  31.423232247  23.421239452  42.15914904
7        25  63.157243234  51.659359265  77.21422468
8        30 111.720527798  91.904345110 135.80942573
9        35 180.954189310 141.354646374 231.64727491
```

```
10        40 274.783129616 200.553656132 376.48662098
11        45 397.205361735 270.720003178 582.78700332
12        50 552.283890387 352.751512640 864.68090044
```

Note now that these, though they look like numbers, are actually *text*, so if you want to display numbers in non-scientific notation like this, do it at the *very end*, after you have finished any calculations with the numbers.

I mentioned `broom` earlier. We can make a data frame out of the one-line summaries of our three models:

```
bind_rows(glance(volume.1), glance(volume.2), glance(volume.3))
```

```
# A tibble: 3 x 12
  r.squared adj.r.squared  sigma statistic     p.value    df logLik   AIC   BIC
      <dbl>         <dbl>  <dbl>     <dbl>       <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.959         0.953 20.4       185.  0.000000822     1 -43.2  92.4  93.4
2     0.953         0.947 21.7       162.  0.00000136      1 -43.8  93.7  94.6
3     0.908         0.896  0.303      78.7 0.0000206       1  -1.12  8.25  9.16
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

(I mistakenly put `glimpse` instead of `glance` there the first time. The former is for a quick look at a *data frame*, while the latter is for a quick look at a *model*.)

The three R-squareds are all high, with the one from the third model being a bit lower as we saw before.

My code is rather repetitious. There has to be a way to streamline it. I was determined to find out how. My solution involves putting the three models in a list-column, and then using `rowwise` to get the `glance` output for each one.

```
tibble(i = 1:3, model = list(volume.1, volume.2, volume.3)) %>%
  rowwise() %>%
  mutate(glances = list(glance(model))) %>%
  unnest(glances)
```

```
# A tibble: 3 x 14
      i model  r.squared adj.r.squared  sigma statistic     p.value    df logLik
  <int> <list>     <dbl>         <dbl>  <dbl>     <dbl>       <dbl> <dbl>  <dbl>
1     1 <lm>       0.959         0.953 20.4       185.  0.000000822     1 -43.2
2     2 <lm>       0.953         0.947 21.7       162.  0.00000136      1 -43.8
3     3 <lm>       0.908         0.896  0.303      78.7 0.0000206       1  -1.12
```

```
# i 5 more variables: AIC <dbl>, BIC <dbl>, deviance <dbl>, df.residual <int>,
#   nobs <int>
```

I almost got caught by forgetting the `list` on the definition of `glances`. I certainly need it, because the output from `glance` is a (one-row) dataframe, not a single number.

It works. You see the three R-squared values in the first column of numbers. The third model is otherwise a lot different from the others because it has a different response variable.

Other thoughts:

How might you measure or estimate the height of a tree (other than by climbing it and dropping a tape measure down)? One way, that works if the tree is fairly isolated, is to walk away from its base. Periodically, you point at the top of the tree, and when the angle between your arm and the ground reaches 45 degrees, you stop walking. (If it's greater than 45 degrees, you walk further away, and if it's less, you walk back towards the tree.) The distance between you and the base of the tree is then equal to the height of the tree, and if you have a long enough tape measure you can measure it.

The above works because the tangent of 45 degrees is 1. If you have a device that will measure the actual angle,[17] you can be any distance away from the tree, point the device at the top, record the angle, and do some trigonometry to estimate the height of the tree (to which you add the height of your eyes).

■

16.22 Tortoise shells and eggs

A biologist measured the length of the carapace (shell) of female tortoises, and then x-rayed the tortoises to count how many eggs they were carrying. The length is measured in millimetres. The data are in link. The biologist is wondering what kind of relationship, if any, there is between the carapace length (as an explanatory variable) and the number of eggs (as a response variable).

(a) Read in the data, and check that your values look reasonable.

Solution

Look at the data first. The columns are aligned and separated by more than one space, so it's `read_table`:

[17] These days, there are apps that will let you do this with your phone. I found one called Clinometer. See also link.

```r
my_url <- "http://ritsokiguess.site/datafiles/tortoise-eggs.txt"
tortoises <- read_table(my_url)
```

```
-- Column specification --------------------------------------------------------
cols(
  length = col_double(),
  eggs = col_double()
)
```

```r
tortoises
```

```
# A tibble: 18 x 2
   length  eggs
    <dbl> <dbl>
 1    284     3
 2    290     2
 3    290     7
 4    290     7
 5    298    11
 6    299    12
 7    302    10
 8    306     8
 9    306     8
10    309     9
11    310    10
12    311    13
13    317     7
14    317     9
15    320     6
16    323    13
17    334     2
18    334     8
```

Those look the same as the values in the data file. (*Some* comment is needed here. I don't much mind what, but something that suggests that you have eyeballed the data and there are no obvious problems: that is what I am looking for.)

■

(b) Obtain a scatterplot, with a smooth trend, of the data.

Solution

Something like this:

```
ggplot(tortoises, aes(x = length, y = eggs)) + geom_point() + geom_smooth()
```

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

■

(c) The biologist expected that a larger tortoise would be able to carry more eggs. Is that what the scatterplot is suggesting? Explain briefly why or why not.

Solution

The biologist's expectation is of an upward trend. But it looks as if the trend on the scatterplot is up, then down, ie. a curve rather than a straight line. So this is not what the biologist was expecting.

■

(d) Fit a straight-line relationship and display the summary.

Solution

```r
tortoises.1 <- lm(eggs ~ length, data = tortoises)
summary(tortoises.1)
```

```
Call:
lm(formula = eggs ~ length, data = tortoises)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.7790 -1.1772 -0.0065  2.0487  4.8556 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.43532   17.34992  -0.025    0.980
length       0.02759    0.05631   0.490    0.631

Residual standard error: 3.411 on 16 degrees of freedom
Multiple R-squared:  0.01478,	Adjusted R-squared:  -0.0468 
F-statistic:  0.24 on 1 and 16 DF,  p-value: 0.6308
```

I didn't ask for a comment, but feel free to observe that this regression is truly awful, with an R-squared of less than 2% and a non-significant effect of `length`.

■

(e) Add a squared term to your regression, fit that and display the summary.

Solution

The `I()` is needed because the raise-to-a-power symbol has a special meaning in a model formula, and we want to *not* use that special meaning:

```r
tortoises.2 <- lm(eggs ~ length + I(length^2), data = tortoises)
summary(tortoises.2)
```

```
Call:
lm(formula = eggs ~ length + I(length^2), data = tortoises)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.0091 -1.8480 -0.1896  2.0989  4.3605 
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -8.999e+02  2.703e+02  -3.329  0.00457 **
length       5.857e+00  1.750e+00   3.347  0.00441 **
I(length^2) -9.425e-03  2.829e-03  -3.332  0.00455 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.671 on 15 degrees of freedom
Multiple R-squared:  0.4338,    Adjusted R-squared:  0.3583
F-statistic: 5.747 on 2 and 15 DF,  p-value: 0.01403
```

Another way is to use `update`:

```r
tortoises.2a <- update(tortoises.1, . ~ . + I(length^2))
summary(tortoises.2a)
```

```
Call:
lm(formula = eggs ~ length + I(length^2), data = tortoises)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0091 -1.8480 -0.1896  2.0989  4.3605

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -8.999e+02  2.703e+02  -3.329  0.00457 **
length       5.857e+00  1.750e+00   3.347  0.00441 **
I(length^2) -9.425e-03  2.829e-03  -3.332  0.00455 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.671 on 15 degrees of freedom
Multiple R-squared:  0.4338,    Adjusted R-squared:  0.3583
F-statistic: 5.747 on 2 and 15 DF,  p-value: 0.01403
```

■

(f) Is a curve better than a line for these data? Justify your answer in two ways: by comparing a measure of fit, and by doing a suitable test of significance.

Solution

An appropriate measure of fit is R-squared. For the straight line, this is about 0.01, and for the regression with the squared term it is about 0.43. This tells us that a straight line fits appallingly badly, and that a curve fits a *lot* better. This doesn't do a test, though. For that, look at the slope of the length-squared term in the second regression; in particular, look at its P-value. This is 0.0045, which is small: the squared term is necessary, and taking it out would be a mistake. The relationship really is curved, and trying to describe it with a straight line would be a big mistake.

■

(g) Make a residual plot for the straight line model: that is, plot the residuals against the fitted values. Does this echo your conclusions of the previous part? In what way? Explain briefly.

Solution

Plot the things called `.fitted` and `.resid` from the regression object, which is not a data frame but you can treat it as if it is for this:

```
ggplot(tortoises.1, aes(x = .fitted, y = .resid)) + geom_point()
```

Up to you whether you put a smooth trend on it or not:

```
ggplot(tortoises.1, aes(x = .fitted, y = .resid)) + geom_point() +
  geom_smooth()
```

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

Looking at the plot, you see a curve, up and down. The most negative residuals go with small or large fitted values; when the fitted value is in the middle, the residual is usually positive. A curve on the residual plot indicates a curve in the actual relationship. We just found above that a curve does fit a lot better, so this is all consistent.

Aside: the grey "envelope" is wide, so there is a lot of scatter on the residual plot. The grey envelope almost contains zero all the way across, so the evidence for a curve (or any other kind of trend) is not all that strong, based on this plot. This is in great contrast to the regression with length-squared, where the length-squared term is *definitely* necessary.

That was all I wanted, but you can certainly look at other plots. Normal quantile plot of the residuals:

```
ggplot(tortoises.1, aes(sample = .resid)) + stat_qq() + stat_qq_line()
```

This is not the best: the low values are a bit too low, so that the whole picture is (a little) skewed to the left.[18]

Another plot you can make is to assess fan-out: you plot the *absolute value*[19] of the residuals against the fitted values. The idea is that if there is fan-out, the absolute value of the residuals will get bigger:

```
ggplot(tortoises.1, aes(x = .fitted, y = abs(.resid))) + geom_point() +
  geom_smooth()
```

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

[18]The very negative residuals are at the left and right of the residual plot; they are there because the relationship is a curve. If you were to look at the residuals from the model with length-squared, you probably wouldn't see this.

[19]The value, but throw away the minus sign if it has one.

I put the smooth curve on as a kind of warning: it looks as if the size of the residuals goes down and then up again as the fitted values increase. But the width of the grey "envelope" and the general scatter of the points suggests that there is really not much happening here at all. On a plot of residuals, the grey envelope is really more informative than the blue smooth trend. On this one, there is no evidence of any fan-out (or fan-in).

■

16.23 Roller coasters

A poll on the Discovery Channel asked people to nominate the best roller-coasters in the United States. We will examine the 10 roller-coasters that received the most votes. Two features of a roller-coaster that are of interest are the distance it drops from start to finish, measured here in feet[20] and the duration of the ride, measured in seconds. Is it true that roller-coasters with a bigger drop also tend to have a longer ride? The data are at link.[21]

(a) Read the data into R and verify that you have a sensible number of rows and columns.

Solution

[20]Roller-coasters work by gravity, so there must be some drop.

[21]These are not to be confused with what your mom insists that you place between your coffee mug and the table.

A `.csv`, so the usual for that:

```
my_url <- "http://ritsokiguess.site/datafiles/coasters.csv"
coasters <- read_csv(my_url)
```

```
Rows: 10 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): coaster_name, state
dbl (2): drop, duration

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
coasters
```

```
# A tibble: 10 x 4
   coaster_name     state         drop duration
   <chr>            <chr>        <dbl>    <dbl>
 1 Incredible Hulk  Florida        105      135
 2 Millennium Force Ohio           300      105
 3 Goliath          California     255      180
 4 Nitro            New Jersey     215      240
 5 Magnum XL-2000   Ohio           195      120
 6 The Beast        Ohio           141       65
 7 Son of Beast     Ohio           214      140
 8 Thunderbolt      Pennsylvania    95       90
 9 Ghost Rider      California     108      160
10 Raven            Indiana         86       90
```

The number of marks for this kind of thing has been decreasing through the course, since by now you ought to have figured out how to do it without looking it up.

There are 10 rows for the promised 10 roller-coasters, and there are several columns: the drop for each roller-coaster and the duration of its ride, as promised, as well as the name of each roller-coaster and the state that it is in. (A lot of them seem to be in Ohio, for some reason that I don't know.) So this all looks good.

■

(b) Make a scatterplot of duration (response) against drop (explanatory), labelling each roller-coaster with its name in such a way that the labels do not overlap. Add a regression line to your plot.

Solution

The last part, about the labels not overlapping, is an invitation to use `ggrepel`, which is the way I'd recommend doing this. (If not, you have to do potentially lots of work organizing where the labels sit relative to the points, which is time you probably don't want to spend.) Thus:

```
library(ggrepel)
ggplot(coasters, aes(x = drop, y = duration, label = coaster_name)) +
  geom_point() + geom_text_repel() +
  geom_smooth(method = "lm", se = FALSE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```

```
Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```

The `se=FALSE` at the end is optional; if you omit it, you get that "envelope" around the line, which is fine here.

Note that with the labelling done this way, you can easily identify which roller-coaster is which.

The warning seems to be `ggplot` being over-zealous; the `geom_point` and the `geom_smooth` don't need a `label`, but `geom_text_repel` certainly does. If it bothers you, move the `label` into the `geom_text_repel`:

```
ggplot(coasters, aes(x = drop, y = duration)) +
  geom_point() + geom_text_repel(aes(label = coaster_name)) +
  geom_smooth(method = "lm", se = FALSE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```

■

(c) Would you say that roller-coasters with a larger drop tend to have a longer ride? Explain briefly.

Solution

I think there are two good answers here: "yes" and "kind of". Supporting "yes" is the fact that the regression line does go uphill, so that overall, or on average, roller-coasters with a larger drop do tend to have a longer duration of ride as well. Supporting "kind of" is the fact that, though the regression line goes uphill, there are a lot of roller-coasters that are some way off the trend, far from the regression line. I am happy to go with either of those. I could also go with "not really" and the same discussion that I attached to "kind of".

■

(d) Find a roller-coaster that is unusual compared to the others. What about its combination of `drop` and `duration` is unusual?

Solution

This is an invitation to find a point that is a long way off the line. I think the obvious choice is my first one below, but I would take either of the others as well:

- "Nitro" is a long way above the line. That means it has a long duration, relative to its drop. There are two other roller-coasters that have a larger drop but not as long a duration. In other words, this roller-coaster drops slowly, presumably by doing a lot of twisting, loop-the-loop and so on.
- "The Beast" is a long way below the line, so it has a short duration relative to its drop. It is actually the shortest ride of all, but is only a bit below average in terms of drop. This suggests that The Beast is one of those rides that drops a long way quickly.
- "Millennium Force" has the biggest drop of all, but a shorter-than-average duration. This looks like another ride with a big drop in it.

A roller-coaster that is "unusual" will have a residual that is large in size (either positive, like Nitro, or negative, like the other two). I didn't ask you to find the residuals, but if you want to, `augment` from `broom` is the smoothest way to go:

```
library(broom)
duration.1 <- lm(duration ~ drop, data = coasters)
augment(duration.1, coasters) %>%
  select(coaster_name, duration, drop, .resid) %>%
  arrange(desc(abs(.resid)))
```

```
# A tibble: 10 x 4
   coaster_name     duration  drop .resid
   <chr>               <dbl> <dbl>  <dbl>
 1 Nitro                 240   215  97.0
 2 The Beast              65   141 -60.1
 3 Millennium Force      105   300 -58.6
```

```
 4 Ghost Rider           160   108  42.8
 5 Goliath               180   255  27.3
 6 Thunderbolt            90    95 -24.0
 7 Raven                  90    86 -21.8
 8 Incredible Hulk       135   105  18.6
 9 Magnum XL-2000        120   195 -18.2
10 Son of Beast          140   214  -2.81
```

`augment` produces a data frame (of the original data frame with some new columns that come from the regression), so I can feed it into a pipe to do things with it, like only displaying the columns I want, and arranging them in order by absolute value of residual, so that the roller-coasters further from the line come out first. This identifies the three that we found above. The fourth one, "Ghost Rider", is like Nitro in that it takes a (relatively) long time to fall not very far. You can also put `augment` in the *middle* of a pipe. What you may have to do then is supply the *original* data frame name to `augment` so that you have everything:

```
coasters %>%
  lm(duration ~ drop, data = .) %>%
  augment(coasters) %>%
  arrange(desc(abs(.resid)))
```

```
# A tibble: 10 x 10
   coaster_name     state      drop duration .fitted .resid  .hat .sigma .cooksd
   <chr>            <chr>     <dbl>    <dbl>   <dbl>  <dbl> <dbl>  <dbl>   <dbl>
 1 Nitro            New Jers~   215      240    143.  97.0  0.138   37.5 3.36e-1
 2 The Beast        Ohio        141       65    125. -60.1  0.118   48.8 1.06e-1
 3 Millennium Force Ohio        300      105    164. -58.6  0.429   45.9 8.70e-1
 4 Ghost Rider      Californ~   108      160    117.  42.8  0.180   51.5 9.46e-2
 5 Goliath          Californ~   255      180    153.  27.3  0.239   53.2 5.91e-2
 6 Thunderbolt      Pennsylv~    95       90    114. -24.0  0.216   53.5 3.91e-2
 7 Raven            Indiana      86       90    112. -21.8  0.245   53.6 3.95e-2
 8 Incredible Hulk  Florida     105      135    116.  18.6  0.188   53.9 1.89e-2
 9 Magnum XL-2000   Ohio        195      120    138. -18.2  0.111   54.0 8.98e-3
10 Son of Beast     Ohio        214      140    143.  -2.81 0.136   54.5 2.77e-4
# i 1 more variable: .std.resid <dbl>
```

I wanted to hang on to the roller-coaster names, so I added the data frame name to `augment`. If you don't (that is, you just put `augment()` in the middle of a pipe), then `augment` "attempts to reconstruct the data from the model".[22] That means you wouldn't get *everything* from the

[22] A quote from the package vignette.

original data frame; you would just get the things that were in the regression. In this case, that means you would lose the coaster names.

A technicality (but one that you should probably care about): `augment` takes up to *two* inputs: a fitted model object like my `duration.1`, and an optional data frame to include other things from, like the coaster names. I had only one input to it in the pipe because the implied first input was the output from the `lm`, which doesn't have a name; the input `coasters` in the pipe was what would normally be the *second* input to `augment`.

■

16.24 Running and blood sugar

A diabetic wants to know how aerobic exercise affects his blood sugar. When his blood sugar reaches 170 (mg/dl), he goes out for a run at a pace of 10 minutes per mile. He runs different distances on different days. Each time he runs, he measures his blood sugar after the run. (The preferred blood sugar level is between 80 and 120 on this scale.) The data are in the file link. Our aim is to predict blood sugar from distance.

(a) Read in the data and display the data frame that you read in.

Solution

From the URL is easiest. These are delimited by one space, as you can tell by looking at the file:

```
my_url <- "http://ritsokiguess.site/datafiles/runner.txt"
runs <- read_delim(my_url, " ")
```

```
Rows: 12 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): distance, blood_sugar

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
runs
```

```
# A tibble: 12 x 2
   distance blood_sugar
      <dbl>       <dbl>
 1      2           136
 2      2           146
 3      2.5         131
 4      2.5         125
 5      3           120
 6      3           116
 7      3.5         104
 8      3.5          95
 9      4            85
10      4            94
11      4.5          83
12      4.5          75
```

That looks like my data file.

■

(b) Make a scatterplot and add a smooth trend to it.

Solution

```
ggplot(runs, aes(x = distance, y = blood_sugar)) + geom_point() +
  geom_smooth()
```

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

`blood_sugar` should be on the vertical axis, since this is what we are trying to predict. Getting the `x` and the `y` right is easy on these, because they are the x and y for your plot.

■

(c) Would you say that the relationship between blood sugar and running distance is approximately linear, or not? It is therefore reasonable to use a regression of blood sugar on distance? Explain briefly.

Solution

I'd say that this is about as linear as you could ever wish for. Neither the pattern of points nor the smooth trend have any kind of noticeable bend in them. (Observing a lack of curvature in either the points or the smooth trend is enough.) The trend is a linear one, so using a regression will be just fine. (If it weren't, the rest of the question would be kind of dumb.)

■

(d) Fit a suitable regression, and obtain the regression output.

Solution

Two steps: `lm` and then `summary`:

```
runs.1 <- lm(blood_sugar ~ distance, data = runs)
summary(runs.1)
```

```
Call:
lm(formula = blood_sugar ~ distance, data = runs)

Residuals:
    Min      1Q  Median      3Q     Max 
-7.8238 -3.6167  0.8333  4.0190  5.5476 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  191.624      5.439   35.23 8.05e-12 ***
distance     -25.371      1.618  -15.68 2.29e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.788 on 10 degrees of freedom
Multiple R-squared:  0.9609,    Adjusted R-squared:  0.957 
F-statistic: 245.7 on 1 and 10 DF,  p-value: 2.287e-08
```

■

(e) How would you *interpret* the slope? That is, what is the slope, and what does that mean about blood sugar and running distance?

Solution

The slope is -25.37. This means that for each additional mile run, the runner's blood sugar will decrease on average by about 25 units.

You can check this from the scatterplot. For example, from 2 to 3 miles, average blood sugar decreases from about 140 to about 115, a drop of 25.

■

(f) Is there a (statistically) significant relationship between running distance and blood sugar? How do you know? Do you find this surprising, given what you have seen so far? Explain briefly.

Solution

Look at the P-value either on the `distance` line (for its t-test) or for the F-statistic on the bottom line. These are the same: 0.000000023. (They will be the same any time there is one x-variable.) This P-value is *way* smaller than 0.05, so there *is* a significant relationship between running distance and blood sugar. This does not surprise me in the slightest, because

the trend on the scatterplot is *so* clear, there's no way it could have happened by chance if in fact there were no relationship between running distance and blood sugar.

■

(g) This diabetic is planning to go for a 3-mile run tomorrow and a 5-mile run the day after. Obtain suitable 95% intervals that say what his blood sugar might be after each of these runs.

Solution

This is a prediction interval, in each case, since we are talking about *individual* runs of 3 miles and 5 miles (not the mean blood sugar after *all* runs of 3 miles, which is what a confidence interval for the mean response would be). The procedure is to set up a data frame with the two `distance` values in it, and then feed that and the regression object into `predict`, coming up in a moment.

```
dists <- c(3, 5)
new <- tibble(distance = dists)
new
```

```
# A tibble: 2 x 1
  distance
     <dbl>
1        3
2        5
```

The important thing is that the name of the column of the new data frame must be *exactly* the same as the name of the explanatory variable in the regression. If they don't match, `predict` won't work. At least, it won't work properly.[23]

If your first thought is `datagrid`, well, that will also work:

```
new2 <- datagrid(model = runs.1, distance = c(5, 10))
new2
```

```
  blood_sugar distance
1    109.1667        5
2    109.1667       10
```

[23]It won't give you an error, but it will go back to the *original* data frame to get distances to predict from, and you will get very confused. This is another example of (base) R trying to make life easier for you, but when it fails, it *fails*.

Use whichever of these methods comes to your mind.

Then, `predict`, because you want prediction intervals rather than confidence intervals for the mean response (which is what `marginaleffects` gives you):

```
pp <- predict(runs.1, new, interval = "p")
pp
```

```
        fit       lwr       upr
1 115.50952 104.37000 126.64905
2  64.76667  51.99545  77.53788
```

and display this with the distances by the side:

```
cbind(new, pp)
```

```
  distance       fit       lwr       upr
1        3 115.50952 104.37000 126.64905
2        5  64.76667  51.99545  77.53788
```

or

```
data.frame(new, pp)
```

```
  distance       fit       lwr       upr
1        3 115.50952 104.37000 126.64905
2        5  64.76667  51.99545  77.53788
```

Blood sugar after a 3-mile run is predicted to be between 104 and 127; after a 5-mile run it is predicted to be between 52 and 77.5.

Extra: both `cbind` and `data.frame` are "base R" ways of combining a data frame with something else to make a new data frame. They are not from the `tidyverse`. The `tidyverse` way is via `tibble` or `bind_cols`, but they are a bit more particular about what they will take: `tibble` takes vectors (single variables) and `bind_cols` takes vectors or data frames. The problem here is that `pp` is not either of those:

```
class(pp)
```

```
[1] "matrix" "array"
```

so that we have to use `as_tibble` first to turn it into a data frame, and thus:

```
pp %>% as_tibble() %>% bind_cols(new)
```

```
# A tibble: 2 x 4
    fit   lwr   upr distance
  <dbl> <dbl> <dbl>    <dbl>
1 116.  104.  127.         3
2  64.8  52.0  77.5        5
```

which puts things backwards, unless you do it like this:

```
new %>% bind_cols(as_tibble(pp))
```

```
# A tibble: 2 x 4
  distance   fit   lwr   upr
     <dbl> <dbl> <dbl> <dbl>
1        3 116.  104.  127.
2        5  64.8  52.0  77.5
```

which is a pretty result from very ugly code.

I also remembered that if you finish with a `select`, you get the columns in the order they were in the `select`:

```
pp %>%
  as_tibble() %>%
  bind_cols(new) %>%
  select(c(distance, everything()))
```

```
# A tibble: 2 x 4
  distance   fit   lwr   upr
     <dbl> <dbl> <dbl> <dbl>
1        3 116.  104.  127.
2        5  64.8  52.0  77.5
```

`everything` is a so-called "select helper". It means "everything except any columns you already named", so this whole thing has the effect of listing the columns with `distance` first and all the other columns afterwards, in the order that they were in before.

■

(h) Which of your two intervals is longer? Does this make sense? Explain briefly.

Solution

The intervals are about 22.25 and 25.5 units long. The one for a 5-mile run is a bit longer. I think this makes sense because 3 miles is close to the average run distance, so there is a lot of "nearby" data. 5 miles is actually longer than any of the runs that were actually done (and therefore we are actually extrapolating), but the important point for the prediction interval is that there is less nearby data: those 2-mile runs don't help so much in predicting blood sugar after a 5-mile run. (They help *some*, because the trend is so linear. This is why the 5-mile interval is not *so* much longer. If the trend were less clear, the 5-mile interval would be more noticeably worse.)

■

16.25 Calories and fat in pizza

The file at link came from a spreadsheet of information about 24 brands of pizza: specifically, per 5-ounce serving, the number of calories, the grams of fat, and the cost (in US dollars). The names of the pizza brands are quite long. This file may open in a spreadsheet when you browse to the link, depending on your computer's setup.

(a) Read in the data and display at least some of the data frame. Are the variables of the right types? (In particular, why is the number of calories labelled one way and the cost labelled a different way?)

Solution

`read_csv` is the thing this time:

```
my_url <- "http://ritsokiguess.site/datafiles/Pizza.csv"
pizza <- read_csv(my_url)
```

```
Rows: 24 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Type
dbl (3): Calories, Fat, Cost

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
pizza
```

```
# A tibble: 24 x 4
   Type                                                  Calories   Fat  Cost
   <chr>                                                    <dbl> <dbl> <dbl>
 1 Domino's Deep Dish with Pepperoni                          385  19.5  1.87
 2 Pizza Hut's Stuffed Crust with Pepperoni                   370  15    1.83
 3 Pizza Hut's Pan Pizza with Pepperoni                       280  14    1.83
 4 Domino's Hand-Tossed with Pepperoni                        305  12    1.67
 5 Pizza Hut's Hand-Tossed with Pepperoni                     230   9    1.63
 6 Little Caesars' Deep Dish with Pepperoni                   350  14.2  1.06
 7 Little Caesars' Original Round with Pepperoni              230   8    0.81
 8 Freschetta Bakes & Rises  4-Cheese                         364  15    0.98
 9 Freschetta Bakes & Rises Sauce Stuffed Crust 4-Cheese      334  11    1.23
10 DiGiorno Rising Crust Four Cheese                          332  12    0.94
# i 14 more rows
```

The four variables are: the brand of pizza, which got read in as text, the number of calories (an integer), and the fat and cost, which are both decimal numbers so they get labelled `dbl`, which is short for "double-precision floating point number".

Anyway, these are apparently the right thing.

Extra: I wanted to mention something else that I discovered yesterday.[24] There is a package called `rio` that will read (and write) data in a whole bunch of different formats in a unified way.[25] Anyway, the usual installation thing, done once:

```r
install.packages("rio")
```

which takes a moment since it probably has to install some other packages too, and then you read in a file like this:

```r
library(rio)
pizza3 <- import(my_url)
head(pizza3)
```

[24] R is like that: sometimes it seems as if it has infinite depth.

[25] It does this by figuring what kind of thing you have, from the extension to its filename, and then calling an appropriate function to read in or write out the data. This is an excellent example of "standing on the shoulders of giants" to make our lives easier. The software does the hard work of figuring out what kind of thing you have and how to read it in; all we do is say `import`.

```
                                      Type Calories  Fat Cost
1        Domino's Deep Dish with Pepperoni      385 19.5 1.87
2 Pizza Hut's Stuffed Crust with Pepperoni      370 15.0 1.83
3     Pizza Hut's Pan Pizza with Pepperoni      280 14.0 1.83
4      Domino's Hand-Tossed with Pepperoni      305 12.0 1.67
5   Pizza Hut's Hand-Tossed with Pepperoni      230  9.0 1.63
6 Little Caesars' Deep Dish with Pepperoni      350 14.2 1.06
```

`import` figures that you have a `.csv` file, so it calls up `read_csv` or similar.

Technical note: `rio` does not use the `read_` functions, so what it gives you is actually a `data.frame` rather than a `tibble`, so that when you display it, you get the whole thing even if it is long. Hence the `head` here and below to display the first six lines.

I originally had the data as an Excel spreadsheet, but `import` will gobble up that pizza too:

```r
my_other_url <- "http://ritsokiguess.site/datafiles/Pizza_E29.xls"
pizza4 <- import(my_other_url)
head(pizza4)
```

```
                                      Type Calories Fat (g) Cost ($)
1        Domino's Deep Dish with Pepperoni      385    19.5     1.87
2 Pizza Hut's Stuffed Crust with Pepperoni      370    15.0     1.83
3     Pizza Hut's Pan Pizza with Pepperoni      280    14.0     1.83
4      Domino's Hand-Tossed with Pepperoni      305    12.0     1.67
5   Pizza Hut's Hand-Tossed with Pepperoni      230     9.0     1.63
6 Little Caesars' Deep Dish with Pepperoni      350    14.2     1.06
```

The corresponding function for writing a data frame to a file in the right format is, predictably enough, called `export`.

■

(b) Make a scatterplot for predicting calories from the number of grams of fat. Add a smooth trend. What kind of relationship do you see, if any?

Solution

All the variable names start with Capital Letters:

```r
ggplot(pizza, aes(x = Fat, y = Calories)) + geom_point() +
  geom_smooth()
```

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

There is definitely an upward trend: the more fat, the more calories. The trend is more or less linear (or, a little bit curved: say what you like, as long as it's not obviously crazy). *I* think, with this much scatter, there's no real justification for fitting a curve.

■

(c) Fit a straight-line relationship, and display the intercept, slope, R-squared, etc. Is there a real relationship between the two variables, or is any apparent trend just chance?

Solution

`lm`, with `summary`:

```r
pizza.1 <- lm(Calories ~ Fat, data = pizza)
summary(pizza.1)
```

```
Call:
lm(formula = Calories ~ Fat, data = pizza)

Residuals:
```

```
   Min     1Q Median     3Q    Max 
-55.44 -11.67   6.18  17.87  41.61 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  194.747     21.605   9.014 7.71e-09 ***
Fat           10.050      1.558   6.449 1.73e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.79 on 22 degrees of freedom
Multiple R-squared:  0.654, Adjusted R-squared:  0.6383 
F-statistic: 41.59 on 1 and 22 DF,  p-value: 1.731e-06
```

To assess whether this trend is real or just chance, look at the P-value on the end of the `Fat` line, or on the bottom line where the F-statistic is (they are the same value of 1.73×10^{-6} or 0.0000017, so you can pick either). This P-value is really small, so the slope is definitely *not* zero, and therefore there really is a relationship between the two variables.

■

(d) Obtain a plot of the residuals against the fitted values for this regression. Does this indicate that there are any problems with this regression, or not? Explain briefly.

Solution

Use the regression object `pizza.1`:

```
ggplot(pizza.1, aes(x = .fitted, y = .resid)) + geom_point() + geom_smooth()
```

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

On my residual plot, I see a slight curve in the smooth trend, but I am not worried about that because the residuals on the plot are all over the place in a seemingly random pattern (the grey envelope is wide and that is pretty close to going straight across). So I think a straight line model is satisfactory.

That's all you needed, but it is also worth looking at a normal quantile plot of the residuals:

```
ggplot(pizza.1, aes(sample = .resid)) + stat_qq() + stat_qq_line()
```

A bit skewed to the left (the low ones are too low).

Also a plot of the absolute residuals, for assessing fan-out:

```
ggplot(pizza.1, aes(x = .fitted, y = abs(.resid))) + geom_point() + geom_smooth()
```

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

A tiny bit of fan-in (residuals getting *smaller* in size as the fitted value gets bigger), but nothing much, I think.

Another way of assessing curvedness is to fit a squared term anyway, and see whether it is significant:

```
pizza.2 <- update(pizza.1, . ~ . + I(Fat^2))
summary(pizza.2)
```

```
Call:
lm(formula = Calories ~ Fat + I(Fat^2), data = pizza)

Residuals:
    Min      1Q  Median      3Q     Max 
-62.103 -14.280   5.513  15.423  35.474 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  90.2544    77.8156   1.160   0.2591  
Fat          25.9717    11.5121   2.256   0.0349 *
I(Fat^2)     -0.5702     0.4086  -1.395   0.1775  
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.25 on 21 degrees of freedom
Multiple R-squared:  0.6834,    Adjusted R-squared:  0.6532
F-statistic: 22.66 on 2 and 21 DF,  p-value: 5.698e-06
```

The fat-squared term is not significant, so that curve on the smooth trend in the (first) residual plot was indeed nothing to get excited about.

■

(e) The research assistant in this study returns with two new brands of pizza (ones that were not in the original data). The fat content of a 5-ounce serving was 12 grams for the first brand and 20 grams for the second brand. For each of these brands of pizza, obtain a suitable 95% interval for the number of calories contained in a 5-ounce serving.

Solution

The suitable interval here is a prediction interval, because we are interested in each case in the calorie content of the *particular* pizza brands that the research assistant returned with (and not, for example, in the mean calorie content for *all* brands of pizza that have 12 grams of fat per serving). Thus:

```r
newfat <- c(12, 20)
new <- tibble(Fat = newfat)
new
```

```
# A tibble: 2 x 1
    Fat
  <dbl>
1    12
2    20
```

```r
preds <- predict(pizza.1, new, interval = "p")
cbind(new, preds)
```

```
  Fat      fit      lwr      upr
1  12 315.3447 260.5524 370.1369
2  20 395.7431 337.1850 454.3011
```

Use `datagrid` to make `new` if you like, but it is a very simple dataframe, so there is no obligation to do it that way.

Or, if you like:

```
as_tibble(preds) %>% bind_cols(new) %>% select(Fat, everything())
```

```
# A tibble: 2 x 4
    Fat   fit   lwr   upr
  <dbl> <dbl> <dbl> <dbl>
1    12  315.  261.  370.
2    20  396.  337.  454.
```

For the pizza with 12 grams of fat, the predicted calories are between 261 and 370 with 95% confidence, and for the pizza with 20 grams of fat, the calories are predicted to be between 337 and 454. (You should write down what these intervals are, and not leave the reader to find them in the output.)

(Remember the steps: create a new data frame containing the values to predict for, and then feed that into `predict` along with the model that you want to use to predict with. The variable in the data frame has to be called *precisely* `Fat` with a capital F, otherwise it won't work.)

These intervals are both pretty awful: you get a very weak picture of how many calories per serving the pizza brands in question might contain. This is for two reasons: (i) there was a fair bit of scatter in the original relationship, R-squared being around 65%, and (ii) even if we knew perfectly where the line went (which we don't), there's no guarantee that individual brands of pizza would be on it anyway. (Prediction intervals are always hit by this double whammy, in that individual observations suffer from variability in where the line goes *and* variability around whatever the line is.)

I was expecting, when I put together this question, that the 20-grams-of-fat interval would be noticeably worse, because 20 is farther away from the mean fat content of all the brands. But there isn't much to choose. For the confidence intervals for the mean calories of *all* brands with these fat contents, the picture is clearer:

```
plot_cap(pizza.1, condition = "Fat")
```

A `fat` value of 12 is close to the middle of the data, so the interval is shorter, but a value of 20 is out near the extreme and the interval is noticeably longer.

This part was a fair bit of work for 3 points, so I'm not insisting that you explain your choice of a prediction interval over a confidence interval, but I think it is still a smart thing to do, even purely from a marks point of view, because if you get it wrong for a semi-plausible reason, you might pick up some partial credit. Not pulling out your prediction intervals from your output is a sure way to lose a point, however.

■

16.26 Where should the fire stations be?

In city planning, one major issue is where to locate fire stations. If a city has too many fire stations, it will spend too much on running them, but if it has too few, there may be unnecessary fire damage because the fire trucks take too long to get to the fire.

The first part of a study of this kind of issue is to understand the relationship between the distance from the fire station (measured in miles in our data set) and the amount of fire damage caused (measured in thousands of dollars). A city recorded the fire damage and distance from fire station for 15 residential fires (which you can take as a sample of "all possible residential fires in that city"). The data are in link.

(a) Read in and display the data, verifying that you have the right number of rows and the right columns.

Solution

A quick check of the data reveals that the data values are separated by exactly one space, so:

```
my_url <- "http://ritsokiguess.site/datafiles/fire_damage.txt"
fire <- read_delim(my_url, " ")
```

```
Rows: 15 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: " "
dbl (2): distance, damage

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
fire
```

```
# A tibble: 15 x 2
   distance damage
      <dbl>  <dbl>
 1      3.4   26.2
 2      1.8   17.8
 3      4.6   31.3
 4      2.3   23.1
 5      3.1   27.5
 6      5.5   36
 7      0.7   14.1
 8      3     22.3
 9      2.6   19.6
10      4.3   31.3
11      2.1   24
12      1.1   17.3
13      6.1   43.2
14      4.8   36.4
15      3.8   26.1
```

15 observations (rows), and promised, and a column each of distances and amounts of fire damage, also as promised.

■

(b) * Obtain a 95% confidence interval for the mean fire damage. (There is nothing here from STAD29, and your answer should have nothing to do with distance.)

Solution

I wanted to dissuade you from thinking too hard here. It's just an ordinary one-sample t-test, extracting the interval from it:

```
t.test(fire$damage)
```

```
    One Sample t-test

data:  fire$damage
t = 12.678, df = 14, p-value = 4.605e-09
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.94488 30.88178
sample estimates:
mean of x
 26.41333
```

Or

```
with(fire, t.test(damage))
```

```
    One Sample t-test

data:  damage
t = 12.678, df = 14, p-value = 4.605e-09
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.94488 30.88178
sample estimates:
mean of x
 26.41333
```

Ignore the P-value (it's testing that the mean is the default *zero*, which makes no sense). The confidence interval either way goes from 21.9 to 30.9 (thousand dollars).

■

(c) Draw a scatterplot for predicting the amount of fire damage from the distance from the fire station. Add a smooth trend to your plot.

Solution

We are predicting fire damage, so that goes on the y-axis:

```
ggplot(fire, aes(x = distance, y = damage)) + geom_point() + geom_smooth()
```

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

■

(d) * Is there a relationship between distance from fire station and fire damage? Is it linear or definitely curved? How strong is it? Explain briefly.

Solution

When the distance is larger, the fire damage is definitely larger, so there is clearly a relationship. I would call this one approximately linear: it wiggles a bit, but it is not to my mind obviously curved. I would also call it a strong relationship, since the points are close to the smooth trend.

■

(e) Fit a regression predicting fire damage from distance. How is the R-squared consistent (or inconsistent) with your answer from part~(here)?

Solution

The regression is an ordinary `lm`:

```
damage.1 <- lm(damage ~ distance, data = fire)
summary(damage.1)
```

```
Call:
lm(formula = damage ~ distance, data = fire)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.4682 -1.4705 -0.1311  1.7915  3.3915 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  10.2779     1.4203   7.237 6.59e-06 ***
distance      4.9193     0.3927  12.525 1.25e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.316 on 13 degrees of freedom
Multiple R-squared:  0.9235,	Adjusted R-squared:  0.9176 
F-statistic: 156.9 on 1 and 13 DF,  p-value: 1.248e-08
```

We need to display the results, since we need to see the R-squared in order to say something about it.

R-squared is about 92%, high, indicating a strong and linear relationship. Back in part~(here), I said that the relationship is linear and strong, which is entirely consistent with such an R-squared. (If you said something different previously, say how it does or doesn't square with this kind of R-squared value.)

Points: one for fitting the regression, one for displaying it, and two (at the grader's discretion) for saying what the R-squared is and how it's consistent (or not) with part~(here).

Extra: if you thought the trend was "definitely curved", you would find that a parabola (or some other kind of curve) was definitely better than a straight line. Here's the parabola:

```r
damage.2 <- lm(damage ~ distance + I(distance^2), data = fire)
summary(damage.2)
```

```

Call:
lm(formula = damage ~ distance + I(distance^2), data = fire)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.8856 -1.6915 -0.0179  1.5490  3.6278 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    13.3395     2.5303   5.272 0.000197 ***
distance        2.6400     1.6302   1.619 0.131327    
I(distance^2)   0.3376     0.2349   1.437 0.176215    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.227 on 12 degrees of freedom
Multiple R-squared:  0.9347,	Adjusted R-squared:  0.9238 
F-statistic: 85.91 on 2 and 12 DF,  p-value: 7.742e-08
```

There's no evidence here that a quadratic is better.

Or you might even have thought from the wiggles that it was more like cubic:

```r
damage.3 <- update(damage.2, . ~ . + I(distance^3))
summary(damage.3)
```

```

Call:
lm(formula = damage ~ distance + I(distance^2) + I(distance^3), 
    data = fire)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.2325 -1.8377  0.0322  1.1512  3.1806 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)    10.8466     4.3618   2.487   0.0302 *
distance        5.9555     4.9610   1.200   0.2552
I(distance^2)  -0.8141     1.6409  -0.496   0.6296
I(distance^3)   0.1141     0.1608   0.709   0.4928
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.274 on 11 degrees of freedom
Multiple R-squared:  0.9376,    Adjusted R-squared:  0.9205
F-statistic: 55.07 on 3 and 11 DF,  p-value: 6.507e-07
```

No evidence that a cubic is better; that increase in R-squared up to about 94% is just chance (bearing in mind that adding *any* x, even a useless one, will increase R-squared).

How bendy is the cubic?

```
ggplot(fire, aes(x = distance, y = damage)) + geom_point() +
  geom_smooth(method = "lm") +
  geom_line(data = damage.3, aes(y = .fitted), colour = "red")
```

```
`geom_smooth()` using formula = 'y ~ x'
```

The cubic, in red, does bend a little, but it doesn't do an obvious job of going through the points better than the straight line does. It seems to be mostly swayed by that one observation with damage over 40, and choosing a relationship by how well it fits one point is flimsy at the best of times. So, by Occam's Razor, we go with the line rather than the cubic because it (i) fits equally well, (ii) is simpler.

■

(f) *Obtain a 95% confidence interval for the mean fire damage* for a residence that is 4 miles from the nearest fire station*. (Note the contrast with part~(here).)

Solution

This is a confidence interval for a mean response at a given value of the explanatory variable. This is as opposed to part~(here), which is averaged over *all* distances. So, follow the steps. Make a tiny data frame with this one value of `distance`:

```
new <- datagrid(model = damage.1, distance = 4)
new
```

```
    damage distance
1 26.41333        4
```

and then

```
pp <- predictions(damage.1, newdata = new)
pp
```

```
 Estimate Std. Error    z Pr(>|z|) 2.5 % 97.5 % distance
       30      0.662 45.3   <0.001  28.7   31.3        4

Columns: rowid, estimate, std.error, statistic, p.value, conf.low, conf.high, damage, distanc
```

28.5 to 31.4 (thousand dollars).

(I saved this one because I want to refer to it again later.)

■

(g) Compare the confidence intervals of parts (here) and (here). Specifically, compare their centres and their lengths, and explain briefly why the results make sense.

Solution

Let me just put them side by side for ease of comparison: part~(here) is:

```
t.test(fire$damage)
```

```
    One Sample t-test

data:  fire$damage
t = 12.678, df = 14, p-value = 4.605e-09
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.94488 30.88178
sample estimates:
mean of x
 26.41333
```

and part~(here)'s is

```
pp
```

```
 Estimate Std. Error    z Pr(>|z|) 2.5 % 97.5 % distance
       30      0.662 45.3   <0.001  28.7   31.3        4

Columns: rowid, estimate, std.error, statistic, p.value, conf.low, conf.high, damage, distanc
```

The centre of the interval is higher for the mean damage when the distance is 4. This is because the mean distance is a bit less than 4:

```
fire %>% summarize(m = mean(distance))
```

```
# A tibble: 1 x 1
      m
  <dbl>
1  3.28
```

We know it's an upward trend, so our best guess at the mean damage is higher if the mean distance is higher (in (here), the distance is *always* 4: we're looking at the mean fire damage for *all* residences that are 4 miles from a fire station.)

What about the lengths of the intervals? The one in (here) is about $30.9 - 21.9 = 9$ (thousand dollars) long, but the one in (here) is only $31.4 - 28.5 = 2.9$ long, much shorter. This makes sense because the relationship is a strong one: knowing the distance from the fire station is very useful, because the bigger it is, the bigger the damage going to be, with near certainty. Said differently, if you know the distance, you can estimate the damage accurately. If you don't know the distance (as is the case in (here)), you're averaging over a lot of different distances and thus there is a lot of uncertainty in the amount of fire damage also.

If you have some reasonable discussion of the reason why the centres and lengths of the intervals differ, I'm happy. It doesn't have to be the same as mine.

■

16.27 Making it stop

If you are driving, and you hit the brakes, how far do you travel before coming to a complete stop? Presumably this depends on how fast you are going. Knowing this relationship is important in setting speed limits on roads. For example, on a very bendy road, the speed limit needs to be low, because you cannot see very far ahead, and there could be something just out of sight that you need to stop for.

Data were collected for a typical car and driver, as shown in http://ritsokiguess.site/datafiles/stopping.csv. These are American data, so the speeds are miles per hour and the stopping distances are in feet.

(a) Read in and display (probably all of) the data.

Solution

The usual:

```
my_url <- "http://ritsokiguess.site/datafiles/stopping.csv"
stopping <- read_csv(my_url)
```

```
Rows: 8 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (2): speed, distance

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
stopping
```

```
# A tibble: 8 x 2
  speed distance
  <dbl>    <dbl>
1     0        0
2    10       20
3    20       50
4    30       95
5    40      150
6    50      220
7    60      300
8    70      400
```

There are only eight observations.

■

(b) Make a suitable plot of the data.

Solution

Two quantitative variables means a scatterplot. Stopping distance is the outcome, so that goes on the y-axis:

```r
ggplot(stopping, aes(x=speed, y=distance)) + geom_point()
```

■

(c) Describe any trend you see in your graph.

Solution

It's an upward trend, but not linear: the stopping distance seems to increase faster at higher speeds.

■

(d) Fit a linear regression predicting stopping distance from speed. (You might have some misgivings about doing this, but do it anyway.)

Solution

Having observed a curved relationship, it seems odd to fit a straight line. But we are going to do it anyway and then critique what we have:

```
stopping.1 <- lm(distance~speed, data=stopping)
summary(stopping.1)
```

```
Call:
lm(formula = distance ~ speed, data = stopping)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-32.738 -22.351  -7.738  16.622  47.083 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -44.1667    22.0821   -2.00   0.0924 .  
speed         5.6726     0.5279   10.75 3.84e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 34.21 on 6 degrees of freedom
Multiple R-squared:  0.9506,	Adjusted R-squared:  0.9424 
F-statistic: 115.5 on 1 and 6 DF,  p-value: 3.837e-05
```

Extra: note that R-squared is actually really high. We come back to that later.

■

(e) Plot the residuals against the fitted values for this regression.

Solution

```r
ggplot(stopping.1, aes(x=.fitted, y=.resid)) + geom_point()
```

■

(f) What do you learn from the residual plot? Does that surprise you? Explain briefly.

Solution

The points on the residual plot form a (perfect) curve, so the original relationship was a curve. This is exactly what we saw on the scatterplot, so to me at least, this is no surprise.

(Make sure you say *how you know* that the original relationship was a curve from looking at the residual plot. Joined-up thinking. There are *two* ways we know that the relationship is a curve. Get them both.)

■

(g) What is the actual relationship between stopping distance and speed, according to the physics? See if you can find out. Cite any books or websites that you use: that is, include a link to a website, or give enough information about a book that the grader could find it.

Solution

I searched for "braking distance and speed" and found the first two things below, that seemed to be useful. Later, I was thinking about the fourth point (which came out of my head) and while searching for other things about that, I found the third thing:

- a British road safety website, that says "The braking distance depends on how fast the vehicle was travelling before the brakes were applied, and is proportional to the square of the initial speed."
- the Wikipedia article on braking distance, which gives the actual formula. This is the velocity squared, divided by a constant that depends on the coefficient of friction. (That is why your driving instructor tells you to leave a bigger gap behind the vehicle in front if it is raining, and an even bigger gap if it is icy.)
- an Australian math booklet that talks specifically about braking distance and derives the formula (and the other equations of motion).
- also, if you have done physics, you might remember the equation of motion $v^2 = u^2 + 2as$, where u is the initial velocity, v is the final velocity, a is the acceleration and s is the distance covered. In this case, $v = 0$ (the car is stationary at the end), and so $-u^2/2a = s$. The acceleration is negative (the car is slowing down), so the left side is, despite appearances, positive. There seems to be a standard assumption that deceleration due to braking is constant (the same for all speeds), at least if you are trying to stop a car in a hurry.

These are all saying that we should add a speed-squared term to our regression, and then we will have the relationship exactly right, according to the physics.

Extra: Another way to measure how far you are behind the vehicle in front is time. Many of the British "motorways" (think 400-series highways) were built when I was young, and I remember a TV commercial that said "Only a Fool Breaks the Two Second Rule".[26] In those days (the linked one is from the 1970s),[27] a lot of British drivers were not used to going that fast, or on roads that straight, so this was a way to know how big a gap to leave, so that you had time to take evasive action if needed. The value of the two-second rule is that it works for any speed, and you don't have to remember a bunch of stopping distances. (When I did my (Canadian) driving theory test, I think I worked out and learned a formula for the stopping distances that I could calculate in my head. I didn't have to get very close since the test was multiple-choice.)

■

(h) Fit the relationship that your research indicated (in the previous part) and display the results. Comment briefly on the R-squared value.

Solution

Add a squared term in `speed`:

```
stopping.2 <- lm(distance~speed+I(speed^2), data=stopping)
summary(stopping.2)
```

[26]This is perhaps not a commercial so much as a public safety message.
[27]There are some typical British cars of the era in the commercial.

```
Call:
lm(formula = distance ~ speed + I(speed^2), data = stopping)

Residuals:
       1        2        3        4        5        6        7        8 
-1.04167  0.98214  0.08929  1.27976 -0.44643 -0.08929 -2.64881  1.87500 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.041667   1.429997   0.728    0.499    
speed       1.151786   0.095433  12.069 6.89e-05 ***
I(speed^2)  0.064583   0.001311  49.267 6.51e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.699 on 5 degrees of freedom
Multiple R-squared:  0.9999,	Adjusted R-squared:  0.9999 
F-statistic: 2.462e+04 on 2 and 5 DF,  p-value: 1.039e-10
```

The R-squared now is basically 1, so that the model fits very close to perfectly.

Extra: you probably found in your research that the distance should be just something times speed squared, with no constant or linear term. Here, though, we have a significant linear term as well. That is probably just chance, since the distances in the data look as if they have been rounded off. With more accurate values, I think the linear term would have been closer to zero.

If you want to go literally for the something-times-speed-squared, you can do that. This doesn't quite work:

```r
stopping.3x <- lm(distance~I(speed^2), data=stopping)
summary(stopping.3x)
```

```
Call:
lm(formula = distance ~ I(speed^2), data = stopping)

Residuals:
     Min       1Q   Median       3Q      Max 
-14.7327  -3.4670   0.6761   6.2323   8.4513 

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.732704   4.362859   3.377   0.0149 *
I(speed^2)   0.079796   0.001805  44.218 8.96e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.514 on 6 degrees of freedom
Multiple R-squared:  0.9969,    Adjusted R-squared:  0.9964
F-statistic:  1955 on 1 and 6 DF,  p-value: 8.958e-09
```

because it still has an intercept in it. In R, the intercept is denoted by `1`. It is always included, unless you explicitly remove it. Some odd things start to happen if you remove the intercept, so it is not a good thing to do unless you know what you are doing. The answers here have some good discussion. Having decided that you *are* going to remove the intercept, you can remove it the same way as anything else (see `update` in the multiple regression lecture) with "minus". I haven't shown you this, so if you do it, you will need to cite your source: that is, say where you learned what to do:

```r
stopping.3 <- lm(distance~I(speed^2)-1, data=stopping)
summary(stopping.3)
```

```
Call:
lm(formula = distance ~ I(speed^2) - 1, data = stopping)

Residuals:
     Min       1Q   Median       3Q      Max 
-12.6123  -0.7859  10.5314  15.5314  19.2141 

Coefficients:
           Estimate Std. Error t value Pr(>|t|)    
I(speed^2) 0.084207   0.001963   42.89 9.77e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.42 on 7 degrees of freedom
Multiple R-squared:  0.9962,    Adjusted R-squared:  0.9957 
F-statistic:  1840 on 1 and 7 DF,  p-value: 9.772e-10
```

The R-squared is still extremely high, much higher than for the straight line. The coefficient value, as I said earlier (citing Wikipedia), depends on the coefficient of friction; the stopping

distances you see typically are based on a dry road, so you have to allow extra distance (or time: see above) if the road is not dry.

■

(i) Somebody says to you "if you have a regression with a high R-squared, like 95%, there is no need to look for a better model." How do you respond to this? Explain briefly.

Solution

An example of a regression with an R-squared of 95% is the straight-line fit from earlier in this question. This is an example of a regression that fits well but is not appropriate because it doesn't capture the form of the relationship.

In general, we are saying that no matter how high R-squared is, we might still be able to improve on the model we have. The flip side is that we might not be able to do any better (with another data set) than an R-squared of, say, 30%, because there is a lot of variability that is, as best as we can assess it, random and not explainable by anything.

Using R-squared as a measure of absolute model quality is, thus, a mistake. Or, to say it perhaps more clearly, asking "how high does R-squared have to be to indicate a good fit?" is asking the wrong question. The right thing to do is to concentrate on getting the form of the model right, and thereby get the R-squared as high as we can for that data set (which might be very high, as here, or not high at all).

■

16.28 Predicting height from foot length

Is it possible to estimate the height of a person from the length of their foot? To find out, 33 (male) students had their height and foot length measured. The data are in http://ritsokiguess.site/datafiles/heightfoot.csv.

(a) Read in and display (some of) the data. (If you are having trouble, make sure you have *exactly* the right URL. The correct URL has no spaces or other strange characters in it.)

Solution

The usual:

```
my_url <- "http://ritsokiguess.site/datafiles/heightfoot.csv"
hf <- read_csv(my_url)
```

```
Rows: 33 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (2): height, foot

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
hf
```

```
# A tibble: 33 x 2
   height  foot
    <dbl> <dbl>
 1   66.5  27
 2   73.5  29
 3   70    25.5
 4   71    27.9
 5   73    27
 6   71    26
 7   71    29
 8   69.5  27
 9   73    29
10   71    27
# i 23 more rows
```

Call the data frame whatever you like, but keeping away from the names `height` and `foot` is probably wise, since those are the names of the columns.

There are indeed 33 rows as promised.

Extra: my comment in the question was to help you if you copy-pasted the file URL into R Studio. Depending on your setup, this might have gotten pasted with a space in it, at the point where it is split over two lines. The *best* way to proceed, one that won't run into this problem, is to *right*-click on the URL and select Copy Link Address (or the equivalent on your system), and then it will put the whole URL on the clipboard in one piece, even if it is split over two lines in the original document, so that pasting it will work without problems.

■

(b) Make a suitable plot of the two variables in the data frame.

Solution

They are both quantitative, so a scatter plot is called for:

```r
ggplot(hf, aes(y=height, x=foot)) + geom_point() + geom_smooth()
```

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

I added a smooth trend, or you could just plot the points. (This is better than plotting a regression line at this stage, because we haven't yet thought about whether the trend is straight.)

Now that we've seen the scatterplot, the trend looks more or less straight (but you should take a look at the scatterplot first, with or without smooth trend, before you put a regression line on it). That point top left is a concern, though, which brings us to…

■

(c) Are there any observations not on the trend of the other points? What is unusual about those observations?

Solution

The observation with height greater than 80 at the top of the graph looks like an outlier and does not follow the trend of the rest of the points. Or, this individual is much taller than you would expect for someone with a foot length of 27 inches. Or, this person is over 7 feet tall, which makes little sense as a height. Say something about what makes this person be off the trend.

■

(d) Fit a regression predicting height from foot length, *including* any observations that you identified in the previous part. For that regression, plot the residuals against the fitted values and make a normal quantile plot of the residuals.

Solution

These things. Displaying the `summary` of the regression is optional, but gives the grader an opportunity to check that your work is all right so far.

```
hf.1 <- lm(height~foot, data=hf)
summary(hf.1)
```

```
Call:
lm(formula = height ~ foot, data = hf)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.7491 -1.3901 -0.0310  0.8918 12.9690 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  34.3363     9.9541   3.449 0.001640 ** 
foot          1.3591     0.3581   3.795 0.000643 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.102 on 31 degrees of freedom
Multiple R-squared:  0.3173,	Adjusted R-squared:  0.2952 
F-statistic: 14.41 on 1 and 31 DF,  p-value: 0.0006428
```

```
ggplot(hf.1, aes(x=.fitted, y=.resid)) + geom_point()
```

```
ggplot(hf.1, aes(sample=.resid)) + stat_qq() + stat_qq_line()
```

Note that we did not exclude the off-trend point. Removing points *because they are outliers* is a **bad** idea. This is a good discussion of the issues.

■

(e) Earlier, you identified one or more observations that were off the trend. How does this point or points show up on each of the plots you drew in the previous part?

Solution

On its own at the top in both cases; the large positive residual on the first plot, and the unusually large value at the top right of the normal quantile plot. (You need to say one thing about each graph, or say as I did that the same kind of thing happens on both graphs.)

Extra: in the residuals vs. fitted values, the other residuals show a slight upward trend. This is because the regression line for these data, with the outlier, is pulled (slightly) closer to the outlier and thus slightly further away from the other points, particularly the ones on the left, compared to the same data but with the outlier removed (which you will be seeing shortly). If the unusual point had happened to have an extreme x (foot length) as well, the effect would have been more pronounced.

This is the kind of thing I mean (made-up data):

```
tibble(x = 1:10) %>%
mutate(y = rnorm(10, 10+2*x, 1)) %>%
mutate(y = ifelse(x == 9, 40, y)) -> madeup
madeup
```

```
# A tibble: 10 x 2
       x     y
   <int> <dbl>
 1     1  13.6
 2     2  13.3
 3     3  15.7
 4     4  17.3
 5     5  20.2
 6     6  22.7
 7     7  22.9
 8     8  26.8
 9     9  40
10    10  31.1
```

```
ggplot(madeup, aes(x=x, y=y)) + geom_point() + geom_smooth()
```

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

The second-last point is off a clearly linear trend otherwise (the smooth gets "distracted" by the outlying off-trend point). Fitting a regression anyway and looking at the residual plot gives this:

```
madeup.1 <- lm(y~x, data = madeup)
ggplot(madeup.1, aes(x = .fitted, y = .resid)) + geom_point()
```

This time you see a rather more obvious downward trend in the other residuals. The problem is not with them, but with the one very positive residual, corresponding to the outlier that is way off the trend on the scatterplot.

The residuals in a regression have to add up to zero. If one of them is very positive (as in the one you did and the example I just showed you), at least some of the other residuals have to become more negative to compensate – the ones on the right just above and the ones on the left in the one you did. If you have done STAC67, you will have some kind of sense of why that is: think about the two equations you have to solve to get the estimates of intercept and slope, and how they are related to the residuals. Slide 6 of this shows them; at the least squares estimates, these two partial derivatives both have to be zero, and the things inside the brackets are the residuals.

■

(f) Any data points that concerned you earlier were actually errors. Create and save a new data frame that does not contain any of those data points.

Solution

Find a way to not pick that outlying point. For example, you can choose the observations with height less than 80:

```
hf %>% filter(height<80) -> hfx
hfx
```

```
# A tibble: 32 x 2
   height  foot
    <dbl> <dbl>
 1   66.5  27
 2   73.5  29
 3   70    25.5
 4   71    27.9
 5   73    27
 6   71    26
 7   71    29
 8   69.5  27
 9   73    29
10   71    27
# i 22 more rows
```

Only 32 rows left.

There are many other possibilities. Find one.

■

(g) Run a regression predicting height from foot length for your data set without errors. Obtain a plot of the residuals against fitted values and a normal quantile plot of the residuals for this regression.

Solution

Code-wise, the same as before, but with the new data set:

```r
hf.2 <- lm(height~foot, data=hfx)
summary(hf.2)
```

```
Call:
lm(formula = height ~ foot, data = hfx)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.5097 -1.0158  0.4757  1.1141  3.9951 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  30.1502     6.5411   4.609 7.00e-05 ***
foot          1.4952     0.2351   6.360 5.12e-07 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.029 on 30 degrees of freedom
Multiple R-squared:  0.5741,    Adjusted R-squared:  0.5599
F-statistic: 40.45 on 1 and 30 DF,  p-value: 5.124e-07
```

```
ggplot(hf.2, aes(x=.fitted, y=.resid)) + geom_point()
```

```
ggplot(hf.2, aes(sample=.resid)) + stat_qq() + stat_qq_line()
```

■

(h) Do you see any problems on the plots you drew in the previous part? Explain briefly.

Solution

For myself, I see a random scatter of points on the first plot, and points close to the line on the second one. Thus I don't see any problems at all. I would declare myself happy with the second regression, after removing the outlier. (Remember that we removed the outlier *because it was an error*, not just because it was an outlier. Outliers can be perfectly correct data points, and if they are, they have to be included in the modelling.)

You might have a different point of view, in which case you need to make the case for it. You might see a (very mild) case of fanning out in the first plot, or the two most negative residuals might be a bit too low. These are not really outliers, though, not at least in comparison to what we had before.

Extra: a standard diagnostic for fanning-out is to plot the *absolute values* of the residuals against the fitted values, with a smooth trend. If this looks like an increasing trend, there is fanning-out; a decreasing trend shows fanning-in. The idea is that we want to see whether the residuals are changing in *size* (for example, getting more positive *and* more negative both):

```
ggplot(hf.2, aes(x=.fitted, y=abs(.resid))) + geom_point() + geom_smooth()
```

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

No evidence of fanning-out at all. In fact, the residuals seem to be smallest in size *in the middle*.

Another thing you might think of is to try Box-Cox:

```
boxcox(height~foot, data=hfx)
```

It looks as if the best λ is -1, and we should predict one over height from foot length. But this plot is deceiving, since it doesn't even show the whole confidence interval for λ! We should

zoom out (a lot) more:

```
boxcox(height~foot, data=hfx, lambda = seq(-8, 8, 0.1))
```

This shows that the confidence interval for λ goes from about -7 to almost 5: that is, *any* value of λ in that interval is supported by the data! This very definitely includes the do-nothing $\lambda = 1$, so there is really no support for any kind of transformation.

■

(i) Find a way to plot the data and *both* regression lines on the same plot, in such a way that you can see which regression line is which. If you get help from anything outside the course materials, cite your source(s).

Solution

This is the same idea as with the windmill data, page 22, though this one is a bit easier since everything is linear (no curves).

The easiest way is to use `geom_smooth` twice, once with the original data set, and then on the one with the outlier removed:

```
ggplot(hf, aes(y=height, x=foot)) + geom_point() + geom_smooth(method = "lm", se=F) +
geom_smooth(data=hfx, method="lm", colour="red", se=F)
```

```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```

I would use the original data set as the "base", since we want to plot its points (including the outlier) as well as its line. Then we want to plot just the line for the second data set. This entails using a `data=` in the second `geom_smooth`, to say that we want to get *this* regression line from a different data set, and also entails drawing this line in a different colour (or in some way distinguishing it from the first one). Putting the `colour` *outside* an `aes` is a way to make the *whole line* red. (Compare how you make points different colours according to their value on a third variable.)

This is, I think, the best way to do it. You can mimic the idea that I used for the windmill data:

```
ggplot(hf, aes(y=height, x=foot)) + geom_point() + geom_smooth(method = "lm", se=F) +
geom_line(data=hf.2, aes(y = .fitted))
```

```
`geom_smooth()` using formula = 'y ~ x'
```

but this is not as good, because you don't need to use the trickery with `geom_line`: the second trend is another regression line not a curve, and we know how to draw regression lines with `geom_smooth` without having to actually fit them. (Doing it this way reveals that you are copying without thinking, instead of wondering whether there is a better way to do it.)

The idea of using a different data set in different "layers" of a plot is quite well-known. For example, the idea is the one in here, though used for a different purpose there (plotting different sets of points instead of different lines).

■

(j) Discuss briefly how removing the observation(s) that were errors has changed where the regression line goes, and whether that is what you expected.

Solution

The regression line for the original data (my blue line) is pulled up compared to the one with outliers removed (red).

This is not very surprising, because we know that regression lines tend to get pulled towards outliers. What was surprising to me was that the difference wasn't very big. Even at the low end, where the lines differ the most, the difference in predicted height is only about one inch. Since regression lines are based on means, I would have expected a big outlier to have moved the line a lot more.

Say something about what you expected, and say something insightful about whether that was what you saw.

Extra: the regression lines are very similar, but their R-squared values are not: 32% and 57% respectively. Having a point far from the line has a big (downward) impact on R-squared.

■