Personal Science Cookbook

Practical techniques to apply scientific principles to your own life.

2/8/23

This is a cover page.

Table of contents

1 Prerequisites

This book is intended to offer practical, step-by-step instructions to solve common problems when doing data analysis for Personal Science.

We'll assume some basic tools.

- A spreadsheet like Microsoft Excel
- The programming language R and the associated development environment RStudio

A good introduction to R is Hands On Programming with R

Part I

General Overview

2 Introduction

There's something you want to understand, probably about yourself, maybe something health- or wellness-related, but it might be something about the world around you. The point is that it's a question of deep interest to you, though unlikely in its current form to be of enough interest to involve professionals. You'd like to apply the principles of science — hypothesis, experiment, analysis — but you don't know enough of the mechanics to get started.

In other words, like a hungry person in a kitchen full of ingredients, you need a cookbook of recipes that can explain in a step-by-step, repeatable manner, how to go from the raw data around you to some fully-baked insights. That's the purpose of the Personal Science Cookbook. Each "recipe" is short and self-contained. Some are more complex than others, but none require any tools or knowledge beyond what is explained in the book.

3 The Principles of Personal Science

Personal Science is about empowering normal people to use the tools of science to help themselves in their daily lives.

When the first microchips enabled desktop computers in the 1970s, people were unsure what to call them.

The word "mini-computer" was already taken, referring to a generation of computers that didn't require entire rooms, so the techie engineers who confronted these new machines called them "microcomputers", a moniker that lives on in the name for one of the first software companies of that generation, Microsoft.

Some people called them "hobby computers", because that seemed to be all they were good for. The most influential early gathering of people using them was called the "Homebrew Computer Club". The term "desktop" was gaining traction, and inspired later generations that called them "laptops", but then the most traditional of all computer companies introduced its first "IBM PC", and suddenly the industry had a new term.

It was a "personal computer" because, for the first time, it was cheap enough and easy enough for a single individual to use it by him (or her) self. In contrast to all previous generations of computing, everything about the device was intended to be used by a single individual. Even if the computer was shared, only one person would use it at a time, and all design decisions reflected that: a single keyboard, monitor, one power switch. You didn't need a team of people to set up and care for the device — it was out-of-the-box something that a single person could set up and use.

It's easy to forget how transformative this was at the time. Computers until then were very expensive — many times more than the cost of a car or even a house. You had to be a large organization — a university, a business — to afford one, and even if somebody magically just gave one to you, you'd need a special place to keep it, with highly-trained technicians just to keep it running, and of course even more well-trained engineers and scientists to get it to do anything useful.

A similar situation exists today in science. New discoveries are made in large institutions, by teams of high-trained people with access to large, expensive equipment. The discoveries are discussed and shared by specialists who are followed by a cadre of specialized interpreters —

journalists, educators, clinicians — who decipher the new scientific results into lay language and ultimately into face-to-face interaction with the public. Committees meet to discuss takeaways from the expensive and time-consuming research, reaching conclusions that are considered generally acceptable enough to result in new actionable treatments and suggestions for "normal people".

This gap between the specialists and the general public, like the gap between mainframe computers and PCs, is eroding thanks to technology.

Actually that's not quite true: the *potential* gap between specialists and the general public is eroding. But reality is still different. It's as if PCs had been invented but no software.

The personal computer revolution was about more than simply cheaper devices. The hardware became useful after it spawned an entire industry of dedicated software makers, educational experts, consultants and systems integrators,

Professional science

We all think science is great…

but what do people mean when they say "science"? 1. Wonder (photos of stars, micrographs, etc.) 2. Technology (photos of roman arch, integrated circuit, moon landing) 3. A way of thinking (photos of "amateur" scientists)

It's tempting to assume that the scientific way of thinking is obvious, and maybe even obviously the only way to think rigorously but that's not really true.

Alternatives to the scientific way of thinking: recipes

My definition of science: a predisposition to the assumption that you're wrong, a nasty mischievous inclination to disbelieve things you can't prove.

A core scientific skill is *curiosity*. Always ask "what if…" thinking in hypotheticals

Religion seems like a classic example of unscientific thinking, but even that I'll challenge. What if you're wrong? Is there a way to experiment, test it?

Science is:

- Curiosity
- Skepticism : an unending belief that you are wrong
  - Low interest in credentials … just because you are "certified" doesn't mean you know any more than I do.
- Bias toward experiments

See Roberts (2004) for examples.

Part II

Techniques

4 Getting Started

All of our examples will be based on a common data set. We'll begin by creating it and explaining how it works.

Let's say you are suffering from unexplained headaches that appear somewhat randomly. You suspect they may be associated with something you eat, but you're not sure, so you've been tracking 14 weeks (98 days) worth of your own data in a spreadsheet that looks like this:

```
library(tidyverse, quietly=TRUE)
library(lubridate, quietly=TRUE)
```

```
set.seed(1984)

x <- tibble(date=seq(from = today()-weeks(14),
                     by = "1 day", length.out = 7*14),
            headache = sample(c(TRUE,FALSE), 7*14,
                              prob = c(.05,.95),
                              replace = TRUE))

knitr::kable( head(x) ) %>% kableExtra::kable_styling()
```

```
write_csv(x,"headache-days.csv")
```

You can download a copy of this file here

It's easy to add a few more variables (columns) to the dataframe: (download)

date	headache
2022-07-19	FALSE
2022-07-20	FALSE
2022-07-21	FALSE
2022-07-22	FALSE
2022-07-23	FALSE
2022-07-24	FALSE

date	headache	icecream	z	wine
2022-07-19	FALSE	TRUE	7.56	0
2022-07-20	FALSE	FALSE	7.38	0
2022-07-21	FALSE	FALSE	5.51	0
2022-07-22	FALSE	TRUE	7.60	0
2022-07-23	FALSE	FALSE	8.36	0
2022-07-24	FALSE	FALSE	6.92	0
2022-07-25	FALSE	FALSE	6.32	0
2022-07-26	FALSE	FALSE	7.52	0
2022-07-27	FALSE	FALSE	7.95	0
2022-07-28	FALSE	FALSE	6.99	0

```
z <- function(x){
m = NULL
for(i in 1:14){
  m = c(c(rep(0,6),
              floor(runif(1,min=0,max=3))),
        m)
}

m
}
x <- tibble(date=seq(from = today()-weeks(14),
                     by = "1 day", length.out = 7*14),
            headache = sample(c(TRUE,FALSE), 7*14,
                              prob = c(.05,.95),
                              replace = TRUE),
            icecream = sample(c(TRUE,FALSE), 7*14,
                              prob = c(.10,.90),
                              replace = TRUE),
            z = runif(7*14, min = -2.5, max = .5) + 8,
            wine = z(0))

knitr::kable( head(x,10), digits = 2) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

```
write_csv(x,"headache-variables.csv")
```

- `headache`: a day when I have a headache
- `icecream`: did I eat ice cream that day?

- `wine`: Number of glasses of wine I drank.
- `z`: Number of hours I slept that day.

5 Methods

Before we discuss techniques for how to analyze your data, let's cover a few basic methods that will be useful for all of the example solutions in this book.

5.1 What is a dataframe?

Self-collected data is almost always best represented by a table of the variables you want to study and the values that you collected for each of those variables. The most common type of table is a spreadsheet, which in Personal Science we refer to as a data table or a *data frame*. Abbreviated "dataframe" or often just "df", it's a table of values and variables that always has the same form:

- columns are variables: the parameters you want to study
- rows are observations: each incident of data you collected.

It's important to get in the habit of this row/column approach to data collection because, as you'll see, all of our tools assume that data will come in a data frame format.

5.1.1 How do I read a dataframe?

Although you are probably used to handling data frames in a spreadsheet program like Excel, in this cookbook we'll need to start by reading the data into R.

Solution

Use the Tidyverse `readr` package. Read a CSV-formatted file with the `read_csv` function. Other Tidyverse let you read many other types of data, including Microsoft Excel (XLSX) files with the function `readxl::read_excel()`.

Regardless of where you get the data, you'll want to read it into a dataframe. In this case, we'll save the CSV contents into the dataframe variable `headache_df`.

```
library(tidyverse)
headache_df <- readr::read_csv("headache-variables.csv")
headache_df %>% head() %>% knitr::kable()
```

date	headache	icecream	z	wine
2022-07-19	FALSE	TRUE	7.557071	0
2022-07-20	FALSE	FALSE	7.379434	0
2022-07-21	FALSE	FALSE	5.512368	0
2022-07-22	FALSE	TRUE	7.600135	0
2022-07-23	FALSE	FALSE	8.362155	0
2022-07-24	FALSE	FALSE	6.924651	0

Here we peeked at the first 6 lines using the function `head()` and then sent it to the `knitr::kable()` function to be printed in this nice format.

5.2 Rolling average

A long series of daily numbers becomes unwieldy after a while, so we'd like to summarize them somehow, perhaps as groups of weeks or months.

Problem You want to take the rolling 7-day average of a series of numbers.

Solution use the `rolling()` functions in package `zoo`:

```
library(zoo)

headache_df %>%
    mutate(sleep7A = rollapply(z,
                               7,
                               function(x) {x = mean(x,na.rm = TRUE)},
                               align = 'right',
                               fill = NA)) %>%
  tail() %>% knitr::kable()
```

date	headache	icecream	z	wine	sleep7A
2022-10-19	FALSE	FALSE	8.120216	0	6.899738
2022-10-20	FALSE	FALSE	5.668099	0	6.738935
2022-10-21	FALSE	FALSE	8.093531	0	7.094138
2022-10-22	TRUE	FALSE	7.679254	0	7.308210
2022-10-23	FALSE	FALSE	7.346326	0	7.283955
2022-10-24	FALSE	FALSE	6.019567	2	7.321820

Using the Tidyverse `mutate()` function, we created a new variable `sleep7A` to hold the 7-day

rolling average for our sleep (`Z`) variable.

Problem How do we skip the days in between and summarize just the averages by week?

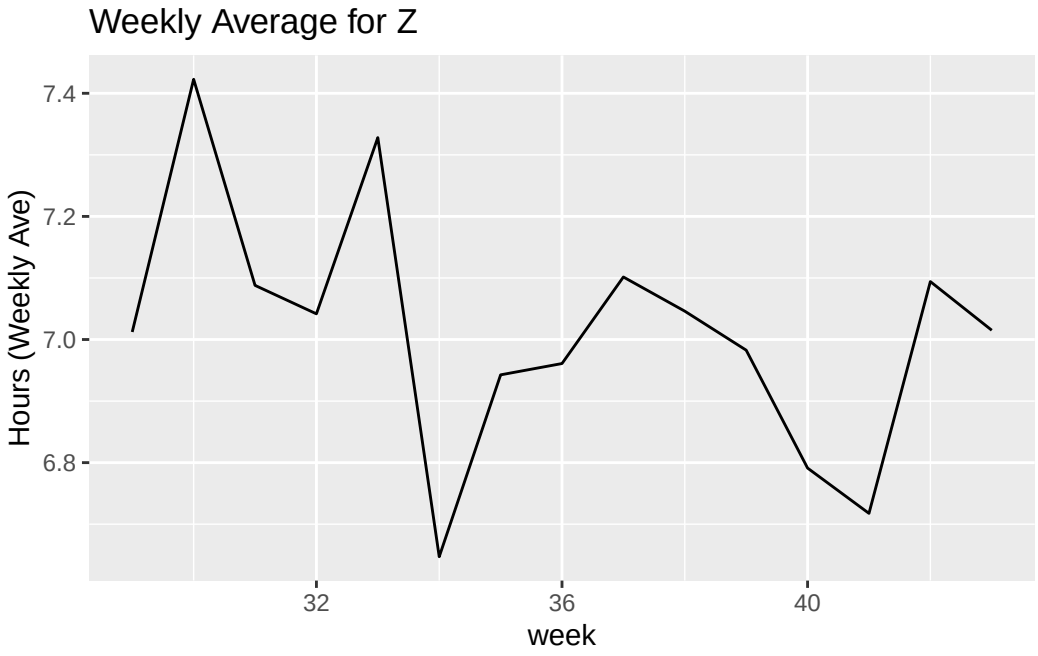
Solution Use `summarize()`.

The Tidyverse function `lubridate::week()` returns the number of complete seven day periods that have occurred between the date and January 1st, plus one.

```
headache_weeks <- headache_df %>%
  mutate(week = lubridate::week(date)) %>%
  group_by(week) %>%
  summarize(week_ave = mean(z))

headache_weeks
```

```
# A tibble: 15 x 2
    week week_ave
   <dbl>    <dbl>
 1    29     7.01
 2    30     7.42
 3    31     7.09
 4    32     7.04
 5    33     7.33
 6    34     6.65
 7    35     6.94
 8    36     6.96
 9    37     7.10
10    38     7.05
11    39     6.98
12    40     6.79
13    41     6.72
14    42     7.09
15    43     7.02
```

```
headache_weeks %>% ggplot(aes(x=week, y = week_ave)) +
  geom_line() +
  labs(title = "Weekly Average for Z", y = "Hours (Weekly Ave)")
```



5.3 Granger Causality

Problem: Given two sets of time series data, x and y, how likely is it that one series will influence the other.

Solution: see Chicken or the Egg? Granger-Causality for the masses

6 Applications

Load the data we created in our Chapter 4 example.

```
x <- readr::read_csv("./headache-variables.csv", show_col_types = FALSE)
```

With my 14 weeks of data, we can do a few basic calculations:

How frequent are my headaches? Simply total the number of headaches and divide by number of days:

```
# headaches per day
sum(x$headache) / length(x$headache)
```

```
[1] 0.08163265
```

6.1 Hypothesis

With the data collected and in a nice dataframe format, we can start to ask what might be driving the headaches. One of the first suspected culprits might be something that I eat.

Based on the data collected so far, can I make any guesses about what might be driving my headaches?

The most obvious place to check is whether I see any patterns on the days when I have headaches. Let's filter for headache days only:

```
x %>% filter(headache)  %>% kableExtra::kable() %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

But maybe the headache takes a day or two to kick in. We can divide the data by week and see if we can spot any patterns in headache frequency:

```
x %>% group_by(week = ntile(date,7)) %>%
  summarise(headaches = sum(headache),
```

date	headache	icecream	z	wine
2022-07-30	TRUE	TRUE	8.478644	0
2022-08-14	TRUE	FALSE	6.880779	0
2022-08-21	TRUE	FALSE	6.909671	0
2022-09-08	TRUE	FALSE	7.278277	0
2022-09-16	TRUE	FALSE	7.797994	0
2022-10-02	TRUE	TRUE	8.385846	0
2022-10-05	TRUE	FALSE	5.849701	0
2022-10-22	TRUE	FALSE	7.679254	0

week	headaches	alcohol	icecream
1	1	0	4
2	1	1	0
3	1	3	0
4	1	2	1
5	1	4	1
6	2	3	3
7	1	3	0

```
            alcohol = sum(wine),
            icecream = sum(icecream))  %>% kableExtra::kable() %>%
kableExtra::kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Buy simply eye-balling the data this way, you might see a pattern. For example, you might spot a week or two with an unusually large number of headaches and notice those weeks are accompanied by an unusually large consumption of some particular food.

But how do you know you're not just guessing? What *looks* like a pattern might be a coincidence. To find out with more certainty, we will apply some statistics.

6.2 T-Testing

Hint: an Excel version of this exercise is in Section 12.1 .

The simplest test is called a "T Test". This is a formula that can compare two equal-sized lists of numbers and return the probability that any differences between the two are the result of chance.

What are the chances that the number of headaches per week is related to the amount of ice cream I eat per week?

week	headaches	alcohol	icecream
1	1	0	4
2	1	1	0
3	1	3	0
4	1	2	1
5	1	4	1
6	2	3	3
7	1	3	0

If there were a relationship between ice cream and headaches each week, I'd expect that over the weeks in this period, the total number of headaches and the total number of ice cream days should be roughly equal.

```
x_week <- x %>% group_by(week = ntile(date,7)) %>%
  summarise(headaches = sum(headache),
            alcohol = sum(wine),
            icecream = sum(icecream))
x_week %>% kableExtra::kable() %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

```
with(x_week, t.test(headaches,icecream))[["p.value"]]
```

```
[1] 0.8254265
```

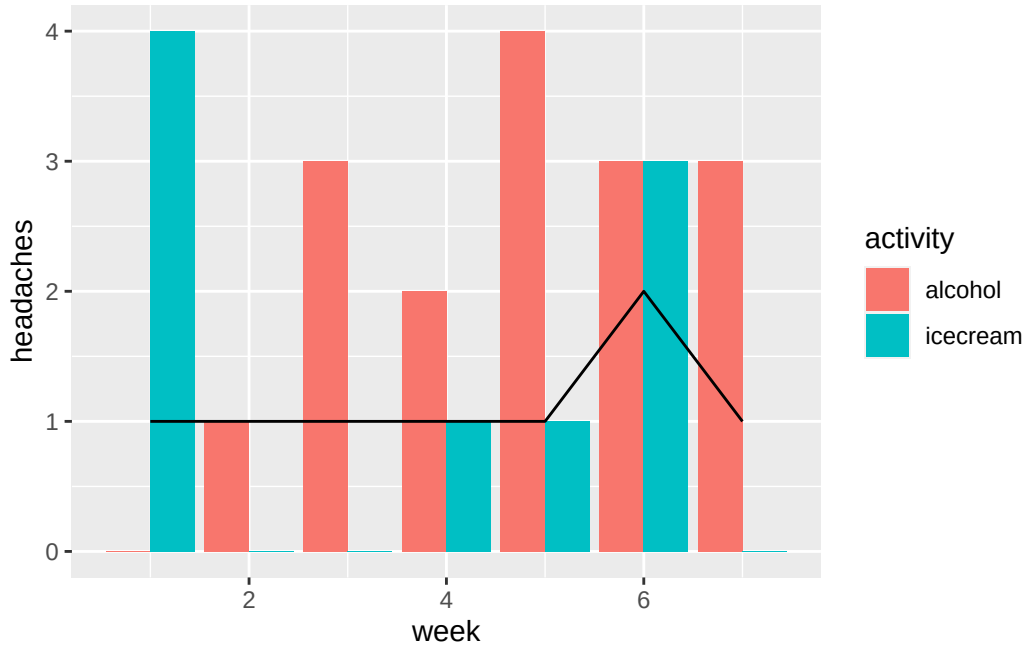
By convention, a p-value less than 0.05 (that is, less than 5\%) is considered statistically significant. While this is not a hard and fast rule, it's often a good place to start. A p-value greater than this is almost certainly due to chance.

6.3 Data visualization

The first step in a more sophisticated analysis is to plot the data to see if we can spot any particular patterns.

```
x_week %>% pivot_longer(names_to = "activity",
                        values_to = "amount",
                        cols = alcohol:icecream ) %>%
  ggplot(aes(x=week,y=headaches)) +
  geom_bar(aes(x=week,y=amount, fill = activity),
           position = "dodge",
```

```
        stat = "identity") +
geom_line(aes(x=week,y=headaches))
```



7 Machine learning

When you are tracking lots of variables, it can be difficult to spot exactly which ones are driving an effect.

In this example, we've fudged the data so that headaches always happen only on days with less than 6 hours of sleep.

Also, headache days always have high stress. But sometimes we have high stress even without a headache.

```
library(tidyverse)
library(lubridate)

set.seed(1984)
WEEKS <- 140

z <- function(x){
m = NULL
for(i in 1:WEEKS){
  m = c(c(rep(0,6),
              floor(runif(1,min=0,max=3))),
        m)
}

m
}
tracking_table <- data.frame(date=seq(from = today()-weeks(WEEKS),
                      by = "1 day", length.out = 7*WEEKS),
            headache = sample(c(TRUE,FALSE), 7*WEEKS,
                              prob = c(.05,.95),
                              replace = TRUE),
            stress = sample(c("low","medium","high"),
                            size = 7*WEEKS,
                            replace = TRUE),
            icecream = sample(c(TRUE,FALSE), 7*WEEKS,
                              prob = c(.10,.90),
```

date	headache	stress	icecream	z	wine
2020-06-03	FALSE	high	FALSE	8.24	0
2020-06-04	FALSE	high	FALSE	6.60	0
2020-06-05	FALSE	medium	TRUE	7.16	0
2020-06-06	FALSE	low	FALSE	6.89	0
2020-06-07	FALSE	high	FALSE	6.04	0
2020-06-08	FALSE	high	FALSE	7.34	0
2020-06-09	TRUE	high	FALSE	5.74	0
2020-06-10	FALSE	high	FALSE	6.87	0
2020-06-11	FALSE	low	FALSE	7.23	0
2020-06-12	FALSE	high	FALSE	6.89	0

```
                        replace = TRUE),
              z = runif(7*WEEKS, min = -2.5, max = .5) + 8,
              wine = z(0))

tracking_table$headache <- tracking_table$z<6  # make headaches on days with low sleep
# tracking_table[tracking_table$stress=="high"] <- "high"  # make stress on headache days

knitr::kable( head(tracking_table,10), digits = 2) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Now make a linear model using the 980 rows from our dataframe.

```
m <- lm(headache~icecream+z+wine+stress, data = tracking_table)
summary(m)
```

```
Call:
lm(formula = headache ~ icecream + z + wine + stress, data = tracking_table)

Residuals:
     Min       1Q   Median       3Q      Max
-0.46762 -0.21394 -0.03199  0.16852  0.57443

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.085871   0.073447  28.400   <2e-16 ***
icecreamTRUE 0.030793   0.028976   1.063    0.288
z           -0.272406   0.010301 -26.445   <2e-16 ***
```

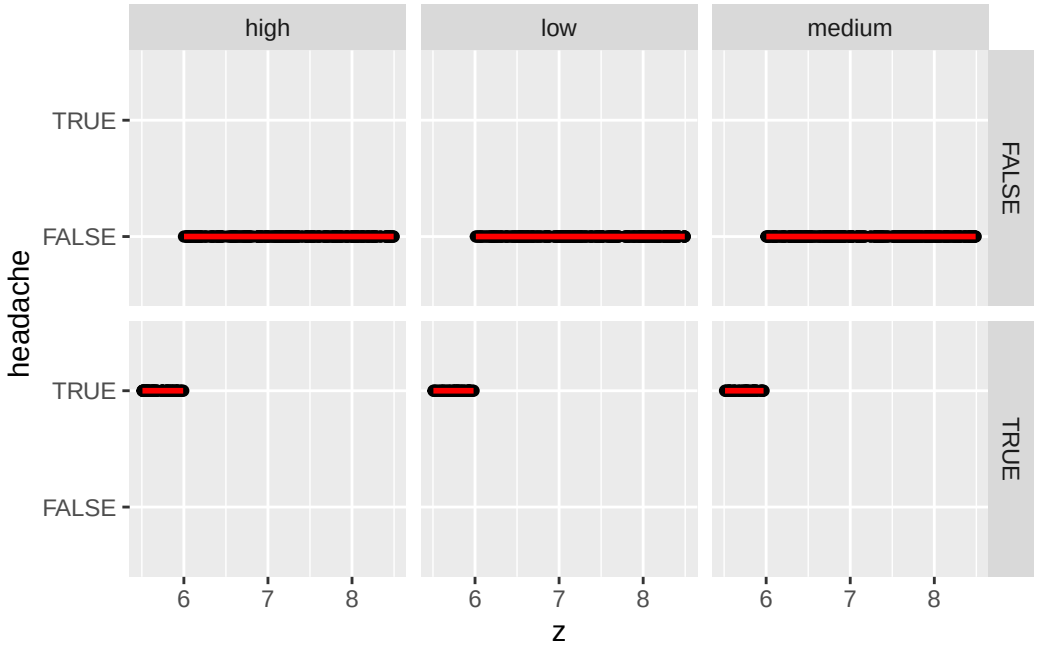
```
wine         -0.002729   0.019588  -0.139    0.889
stresslow    -0.012375   0.022061  -0.561    0.575
stressmedium -0.034324   0.021410  -1.603    0.109
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2776 on 974 degrees of freedom
Multiple R-squared:  0.4216,    Adjusted R-squared:  0.4186
F-statistic:   142 on 5 and 974 DF,  p-value: < 2.2e-16
```

Those triple dots ("***") in the right-hand column for Coefficients indicate items of very high significance.

In this case, as expected, there seems to be a strong relationship between both sleep and stress.

A quick yet dramatic way to visualize this uses the `stat_smooth` function of `ggplot`:

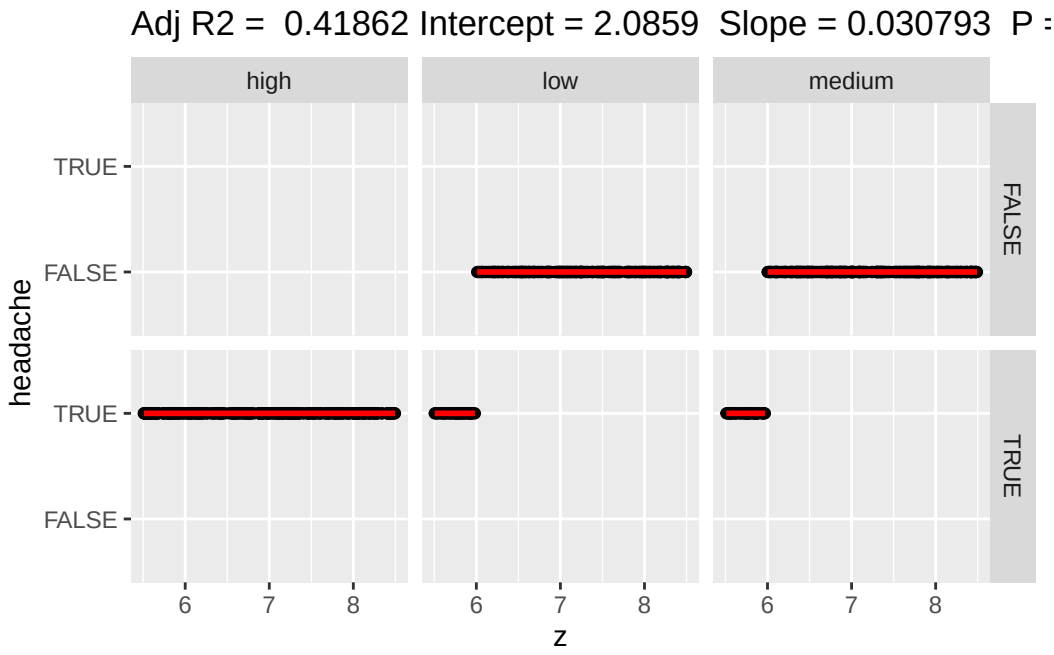
```
tracking_table %>% ggplot(aes(y=headache, x=z)) + geom_point() + stat_smooth(formula = y~x
facet_grid(headache ~ stress)
```



We clearly see that headaches always occur on days when Z<6.

And here's what happens if we change the data so headaches always happen when stress is high.

```
tracking_table[tracking_table$stress == "high",]$headache <- TRUE

tracking_table %>% ggplot(aes(y=headache, x=z)) + geom_point() + stat_smooth(formula = y~x
facet_grid(headache ~ . + stress) +
  labs(title = paste("Adj R2 = ",signif(summary(m)$adj.r.squared, 5),
                     "Intercept =",signif(m$coef[[1]],5 ),
                     " Slope =",signif(m$coef[[2]], 5),
                     " P =",signif(summary(m)$coef[2,4], 5)))
```



As expected, the plot shows headaches either when `z<6` or when stress is high.

8 headaches

Another way to study this data is through machine learning. Instead of a straightforward mathematical transformation of *all* the data, we can take a subset (called the "training" set) and use algorithms to build a model to describe the data. Then we can apply that model to the rest of the data (called the "test" set).

```
library(tidyverse)
library(caret)
```

```
Loading required package: lattice
```

```
Attaching package: 'caret'
```

```
The following object is masked from 'package:purrr':

    lift
```

8.1 Generate the sample data

```
# create a data frame with the generated data
mydata <- tracking_table
mydata$migraine <- if_else(mydata$headache, "yes","no")
mydata$migraine <- factor(mydata$migraine)
mydata$sleep <- mydata$z
```

8.2 Select a random subset for training

```
# select 25% of the rows from the data frame for training

mydata_train <- mydata %>% sample_frac(size=0.25)

# use the remaining rows for testing
mydata_test <- mydata %>% anti_join(mydata_train)
```

```
Joining, by = c("date", "headache", "stress", "icecream", "z", "wine",
"migraine", "sleep")
```

```
response <- "migraine"
```

8.3 Make a prediction model

```
# Convert the response variable to a factor
mydata_train$headache <- as.factor(mydata_train$headache)
mydata_test$headache <- as.factor(mydata_test$headache)

# Train the model using the training data
model_caret <- caret::train(migraine ~ stress + icecream + z + wine,
                      data = mydata_train,
                      method = "glm",
                      family = binomial())
```

```
Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: algorithm did not converge

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

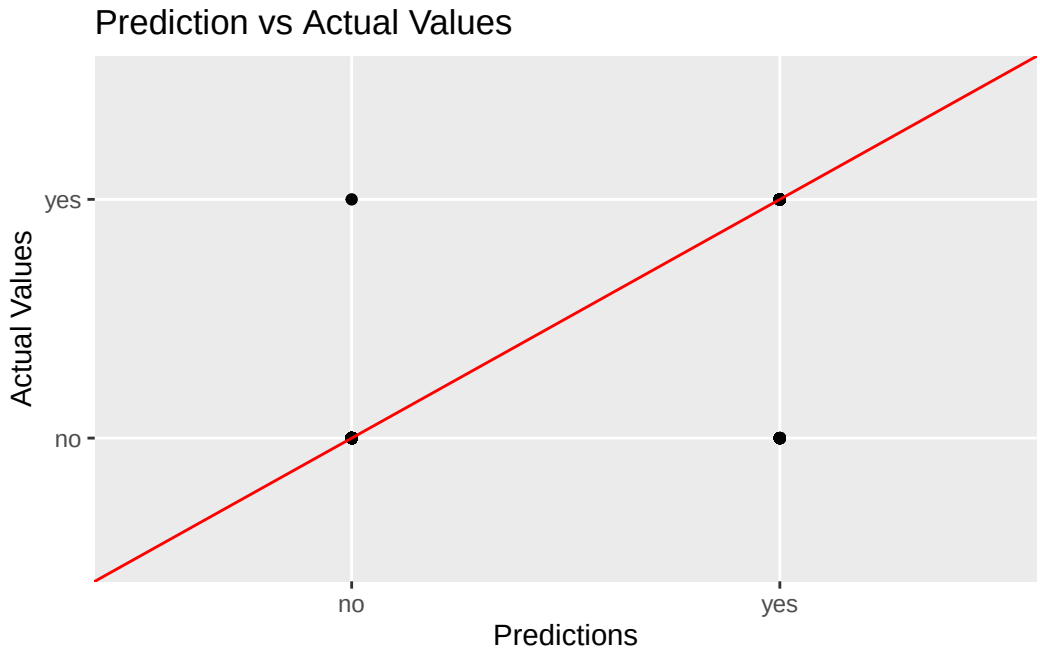
```r
# Make predictions on the test data
predictions_caret <- predict(model_caret, mydata_test)

# Evaluate the model's performance on the test data
performance_caret <- postResample(predictions_caret, mydata_test$migraine)
print(performance_caret)
```

```
 Accuracy     Kappa
0.9768707 0.9121351
```

8.4 Plot the result

```r
# Plot the model's predictions against the actual values
ggplot(mydata_test, aes(x = predictions_caret, y = mydata_test[,response])) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  xlab("Predictions") +
  ylab("Actual Values") +
```

```
ggtitle("Prediction vs Actual Values")
```



9 Handling microbiome data

9.1 A word about microbiome sequencing

When a microbiome sample is sequenced by a genetic sequencing machine, the results are presented in large files, called FASTQ, made of the A, C, T, G letters of the genetic code along with other information about measurement accuracy and more. The final report sent to you as a customer, builds from these files using a bioinformatics "pipeline" designed to summarize the genetic code into a more readable format. Embedded within the pipeline are dozens of assumptions about how to best interpret the genetic letters, including how to handle cases where the interpretation is unclear, or even arbitrary. For example, although the sequence of a common microbe like *Streptococcus mutans* is well-understood, how close does a sequence have to be before the report can confidently describe it as a member of that species? Different pipelines make different assumptions. One might say it can be off by 10 letters, while another might say 5; other pipelines might judge based on the particular microbe. And what should the report do when the sequencer returns less-than-confident results? No sequencer can be perfect all of the time, so by necessity some allowance must be made for how much leeway should be allowed in an interpretation.

9.2 Microbiome datasets are compositional

Once the pipeline has been tweaked to give consistent answers for a particular lab, another question awaits.

Since most tests report the *relative* abundance of a particular microbe, the totals will always sum to 100%. While this makes sense when you want to know the overall composition of the microbiome, it may not be as useful when studying how the results from one day compares to another.

The reason is *compositionality*, sometimes called the "sum to 1" problem. To explain this, let's use a concrete example.

9.3 Example

Suppose we have the following result for our first test:

Test 1		
Microbe	Absolute	Relative
A	100	10%
B	500	50%
C	400	40%
D	0	0%
Total	1000	100%

We don't specify the units in the "Absolute" column, but it can be whatever you like: grams, tons, mg/mL – it doesn't matter. In this simple example, we measure a total of 1000 (of something) and compute the various relative amounts. All is well.

In our second test, for whatever reason, we collect a lot more stuff, leading to a larger *absolute* amount but the relative amounts are unchanged.

Test 2		
Microbe	Absolute	Relative
A	150	10%
B	750	50%
C	600	40%
D	0	0%
Total	1500	100%

But now consider a different case. This time, for some reason one of the three microbes has a massive increase in absolute terms. Importantly, *none of the other microbes changed*. This might happen if your sample were somehow contaminated, for example, perhaps from some extraneous microbe entering the tube after you sampled it. Or it could be that the sampling site suddenly had a new growth of an new microbe that doesn't affect anything else. Lots of reasons could explain why the *absolute* values of various microbes could be unchanged even the *relative* values are substantially different.

Test 3A		
Microbe	Absolute	Relative
A	150	8%
B	750	38%
C	600	30%
D	500	25%

Test 3A		
Total	2000	100%

But you don't need contamination for a slight change in one microbe to have a major impact on the *relative* abundance of the others.

Watch what happens when two microbes, A and B, are unchanged while two others swap abundance amounts.

Test 3B		
Microbe	Absolute	Relative
A	150	8%
B	750	38%
C	700	35%
D	400	20%
Total	2000	100%

A and B appear to have the same relative abundances they did in Test 2. This simple case matches our intuition: we expect that the relative values of A and B would be no different than Test 3A. The absolute totals are the same, so again all is well.

But microbes exist in an *ecology*. They're not independent of one another. Often an increase or decrease in one will drive a corresponding change in another.

Consider the interesting case where one one microbe (A) doubles in abundance, causing another (B) to halve. Although the changes are directly related to one another, it's hard to see that in the type of relative summary we get from our report.

Test 2B		
Microbe	Absolute	Relative
A	200	24%
B	250	29%
C	400	47%
D	0	0%
Total	850	100%

In 2B, a major change happened – the abundance of one microbe (A) exploded and caused another (B) to plunge. Although another, independent microbe (C) was completely unaffected

by this change, when we look only at the relative differences, we might be fooled into thinking that C changed as well, though it didn't.

Which matters more, absolute values or relative ones? To the extent that the microbiome is synthesizing or digesting various metabolites in the body, it's clear that *absolute* values are what we want to watch. But absolute abundances are too hard to track – you'd need to grab the entire microbiome somehow. So instead we assume that the microbiome *as a whole* maintains a roughly constant absolute volume and that the only change is the relative abundances.

Is that true? It seems unlikely. Other living populations rise and fall depending on all sorts of factors. Your backyard garden, for example, doesn't have the same absolute volume from one day to another. If you only knew the *relative* percentage of tomatoes versus cucumbers, would you really know much about your harvest?

9.4 The solution

This problem has been noticed for more than a hundred years in every field touched by statistics: ecology, economics, geology and more. Whenever you have an instrument that can only measure a subset of something, you must make allowances for the fact that the final measure is reported in units of 100%.

The solution is to make calculations based not on overall *percentages*, but on ratios of each component. The statistics are more complicated, but that's the only way to make the final result usable.

9.5 Bottom line

It's very hard to make judgements one way or another from simple comparisons of relative abundance changes from one sample to another. Too many factors determine the measured levels of the various microbes.

Despite this, we know empirically that the overall relative abundances are reasonably stable from one collection to another. Not precisely stable, but at least at the highest, say, phylum levels, the abundances track fairly consistently from day to day. In the oral microbiome, for example, *Streptococcus* is almost always the lead phylum, with *Neissaria* and *Rothia* competing with a few others for second or third place.

Meanwhile, in larger population studies of say thousands of people sampled multiple times, some significant patterns emerge of microbes that are consistently over- or under-represented in various disease states.

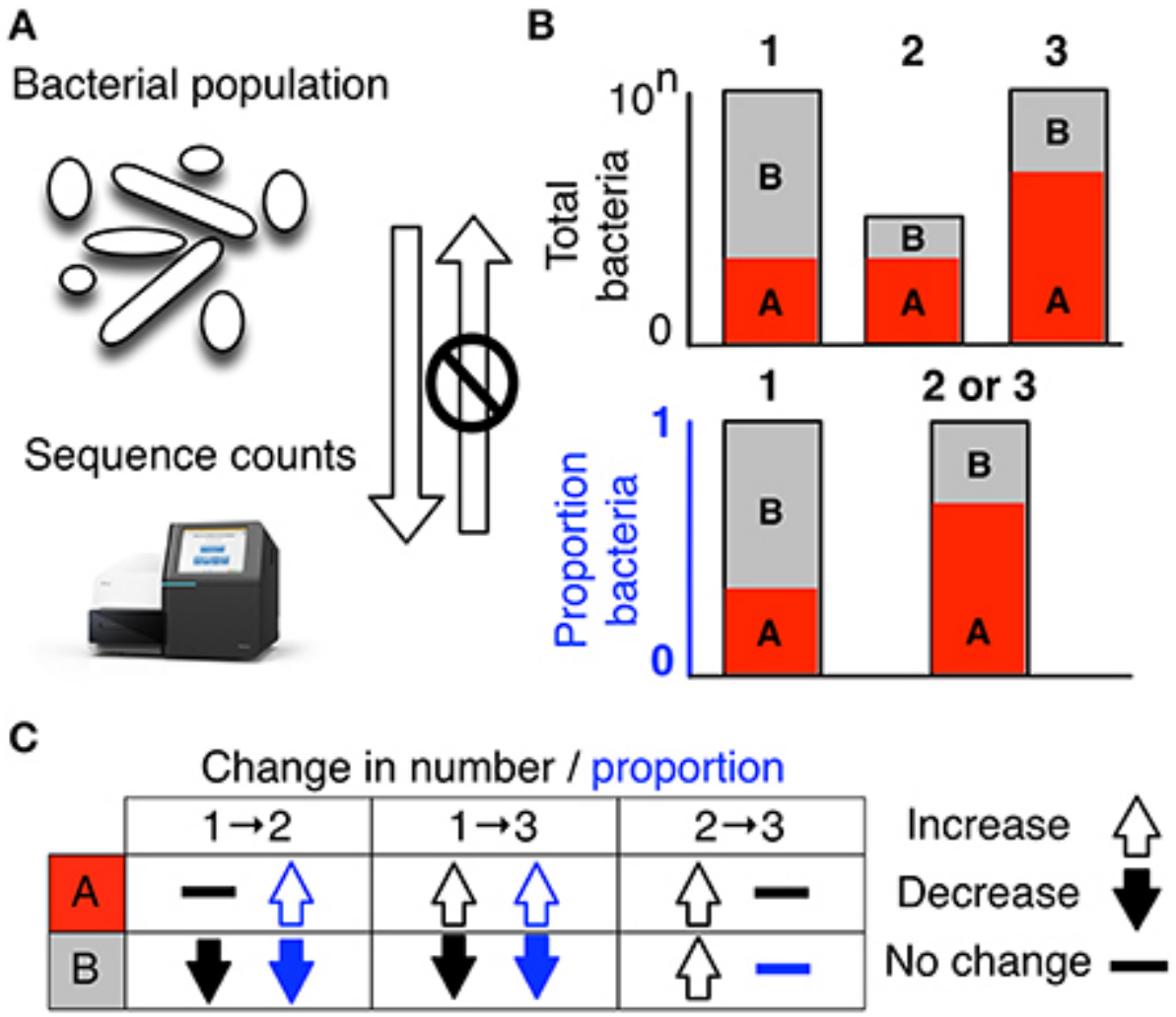


Figure 9.1: FIGURE 1 (from Gloor et al. (2017))) High-throughput sequencing data are compositional. (A) illustrates that the data observed after sequencing a set of nucleic acids from a bacterial population cannot inform on the absolute abundance of molecules. The number of counts in a high throughput sequencing (HTS) dataset reflect the proportion of counts per feature (OTU, gene, etc.) per sample, multiplied by the sequencing depth. Therefore, only the relative abundances are available. The bar plots in (B) show the difference between the count of molecules and the proportion of molecules for two features, A (red) and B (gray) in three samples. The top bar graphs show the total counts for three samples, and the height of the color illustrates the total count of the feature. When the three samples are sequenced we lose the absolute count information and only have relative abundances, proportions, or "normalized counts" as shown in the bottom bar graph. Note that features A and B in samples 2 and 3 appear with the same relative abundances, even though the counts in the environment are different. The table below in (C) shows real and perceived changes for each sample if we transition from one sample to another.

Or consider our garden analogy. Knowing the relative percentage of tomatoes and cucumbers might be useful if we had data meticulously collected from thousands of other backyard gardens, along with some "metadata" about each gardener's assessment of their harvest. You might notice, then, that gardeners unhappy with their tomato crop tend to have lower cucumber yield too. Or there might be a strong correlation between tomato yield and herbicide usage – on average. Still, many or perhaps even most gardens will be significantly different. For example, if the relative abundances appear to match the average, you might be fooled into thinking that a garden suffering from an overall poor harvest is fine.

In other words, treat numbers like "relative abundance" with an appropriate level of skepticism.

Gloor et al. (2017)

10 Averages vs n-of-1

Consider the following case:

1000 people took Vitamin D for 6 months. We measured their Vitamin D levels before and after, and sure enough: the average levels *after* are higher than *before*.

More specifically, let's say the average at the beginning of the study is 30 mg/nL, widely considered the absolute minimum for a healthy person.

```
study <- tibble(subject=1:N,
                vitamind=rnorm(n=N,
                               mean=MEAN_VITAMIN_D,
                               sd=MEAN_VITAMIN_D
                )
                ) %>%
  transmute(subject, vitamind=if_else(vitamind<0, 0,vitamind))

study_plot <- study %>% ggplot(aes(x=subject,y=vitamind)) +
  geom_point() +
  geom_smooth(method= lm, formula= y ~ x, color="red") +
  labs(x="Subject", y = "Vitamin D (ng/mL)", title = "Vitamin D levels in all subjects")

study_plot
```

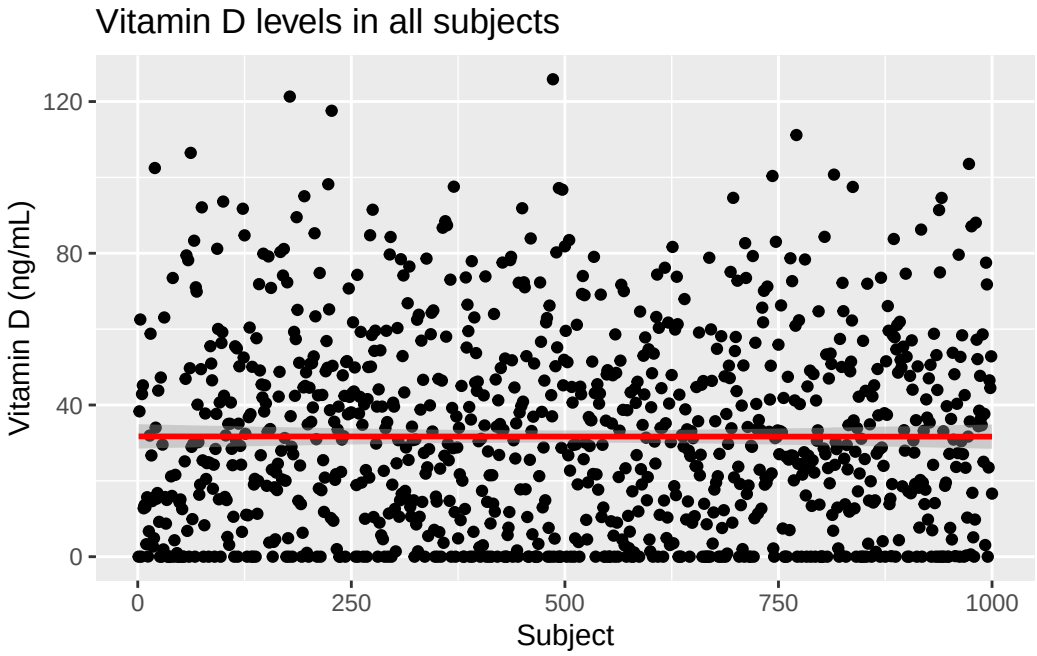


Figure 10.1: Red line indicates the average slope of the points

Note that, although the *average* level is about 30 mg/mL, there are many subjects whose levels are considerable above and below that.

```
maxd <- max(study$vitamind)
study_plot +
  geom_rect(xmin=0,ymin=0,xmax=N, ymax=MEAN_VITAMIN_D - sd(1:MEAN_VITAMIN_D),
            fill = "lightblue",
            alpha = 0.007) +
    geom_rect(xmin=0,ymin=MEAN_VITAMIN_D + sd(1:MEAN_VITAMIN_D),xmax=N, ymax=maxd,
            fill = "lightblue",
            alpha = 0.007)
```

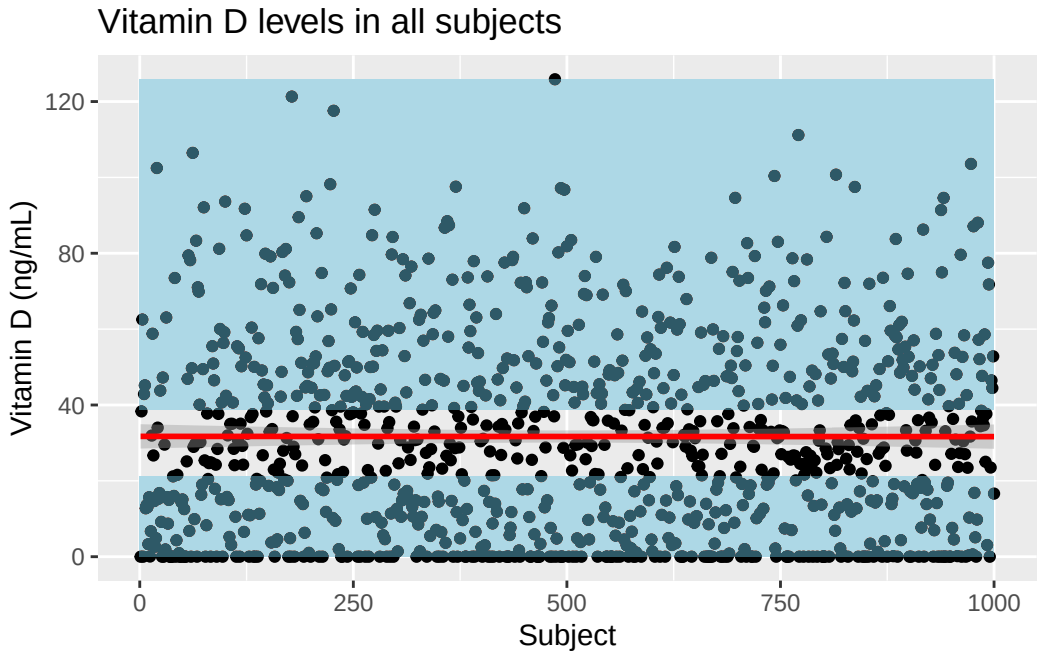


Figure 10.2: Shaded area represents points more than a standard deviation outlier.

Another way to represent the study is with a boxplot variation called a *violin plot*.

```
study %>% ggplot() +
  geom_violin(aes(x=subject,y=vitamind),
              draw_quantiles = c(0.25, 0.5, 0.75)) +
  labs(x="", y = "Vitamin D (ng/mL)", title = "Vitamin D levels in all subjects")
```

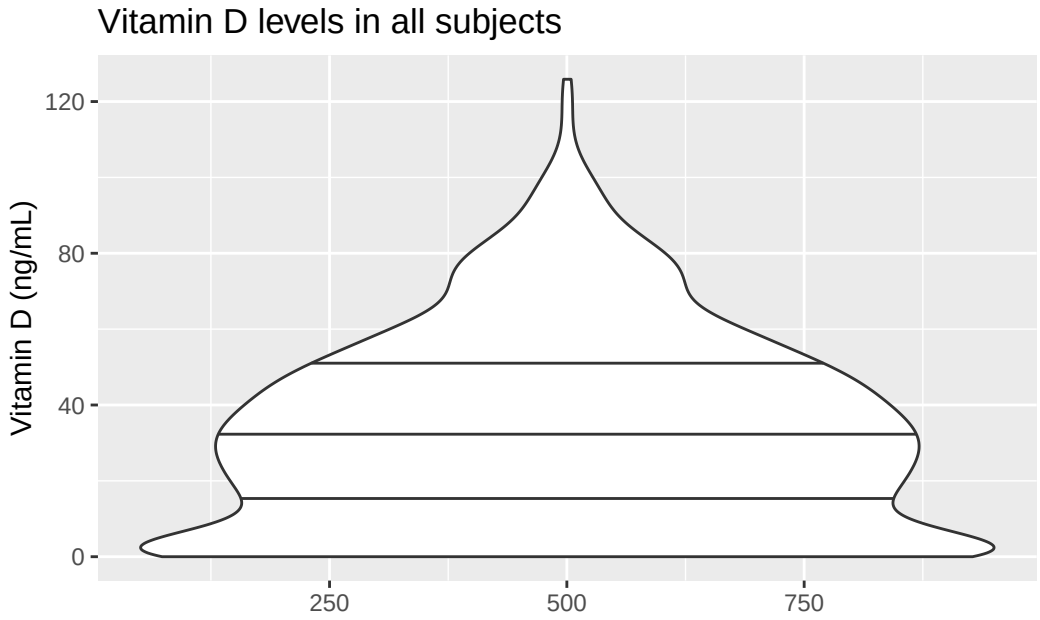


Figure 10.3: The width of this plot shows the proportion of subjects at particular Vitamin D levels. (The horizontal lines indicate the different quartiles of the data)

To make this more fun, let's assume we have additional information about the subjects in our trial.

```
study <- cbind(study,
 nationality = replicate(N, sample(c("USA","Japan","Europe"),size=1)),
 weight = rnorm(n=N, mean = 120, sd = 50))
```

11 Final Words

Download a copy in PDF or ePub.

12 Appendix: Excel Examples

12.1 Is it chance? T-Test

Problem

You tried an intervention and want to see if it worked. How likely is it that the results were chance?

Solution

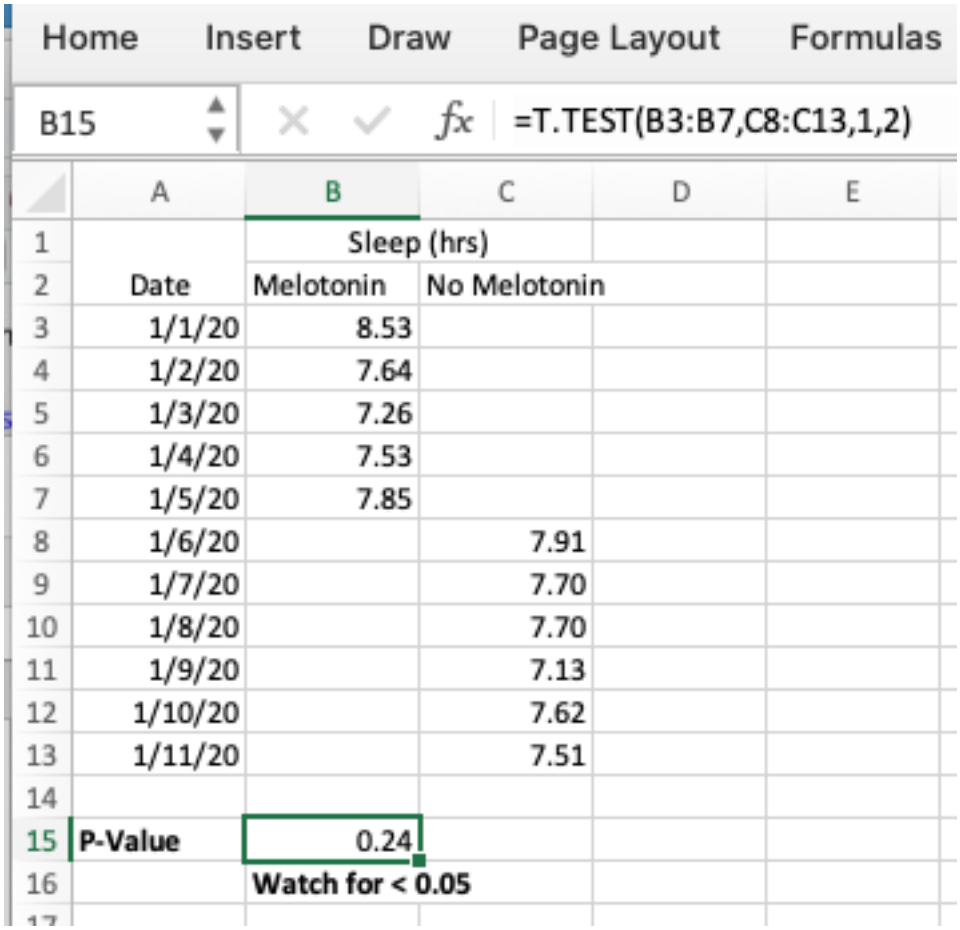
One of the simplest tests is a "T-Test", sometimes called a "Student T Test".

Statisticians use the concept of *P Value* to discuss the how often a result might appear to be significant even when it's not. While this crude measure doesn't describe all the ways something might happen due to chance, generally the lower the P Value, the better. Professional scientists, especially those who understand statistics, will get touchy if you claim a result based purely on P Values, but for Personal Science purposes, it's a good start. There is no "correct" cutoff value that can determine the likelihood that something is due to chance alone, but traditionally people assume that anything under 0.05 deserves a closer look.

Here's an example for how to do this in Excel.

Suppose you'd like to know if taking a melotonin supplement will help you sleep longer. You've measured your daily sleep, taking the supplements on some days (the "intervention") and not on others ("control").

A simple spreadsheet might look like this:



	A	B	C	D	E
1		Sleep (hrs)			
2	Date	Melotonin	No Melotonin		
3	1/1/20	8.53			
4	1/2/20	7.64			
5	1/3/20	7.26			
6	1/4/20	7.53			
7	1/5/20	7.85			
8	1/6/20		7.91		
9	1/7/20		7.70		
10	1/8/20		7.70		
11	1/9/20		7.13		
12	1/10/20		7.62		
13	1/11/20		7.51		
14					
15	**P-Value**	0.24			
16		**Watch for < 0.05**			

Track your sleep under two columns: one for nights when you took the supplement, and the other for nights you didn't.

The built-in Excel statistical function `T.TEST` will calculate the P-Value when you give it two ranges, the "intervention" (nights we took melotonin) and the "control" (nights without).

See the screenshot for the exact formula in this case:

```
=T.TEST(array1,array2,tails,type)
```

Enter a `1` for `tails` (because we're only interested in one measurement, sleep) and a `2` for type (because in this case our samples are not of the same length).

The P Value in this example, `0.24`, is above `0.05` and therefore we will assume that any difference in sleep between the nights is due to pure chance.

13 References

Gloor, Gregory B., Jean M. Macklaim, Vera Pawlowsky-Glahn, and Juan J. Egozcue. 2017. "Microbiome Datasets Are Compositional: And This Is Not Optional." *Frontiers in Microbiology* 8. https://www.frontiersin.org/articles/10.3389/fmicb.2017.02224.

Roberts, Seth. 2004. "Self-Experimentation as a Source of New Ideas: Ten Examples about Sleep, Mood, Health, and Weight." http://www.escholarship.org/uc/item/2xc2h866.