Bootstrapping and Randomisation tests

MI Stefan

Semester 2, 2018/19

Introduction

Learning Objectives

Rattus Binomialis

Problem

A rat is performing a "2-alternative forced choice" (2AFC) task in which it must identify an odor presented at a central port. If it detects odor 'A' it should choose the right-hand port for a reward; if it detects odor 'B' is should choose the other port.

Early in the rat's training, you want ot know whether the rat has learned the task yet. So you decide to do a test and keep track of his correct rate for a block of 50 trials. After 50 trials, we see that the rat has gotten 31 trials correct (19 trials wrong) for an average of 62 percent correct. You want to know if the rat has learned the task or if he is still guessing.

All hypothesis tests are the same . . .

Since we are performing a hypothesis test, we need to ask ourselves what the world would look like if the Null Hypothesis was correct. What is the Null Hypothesis?

Simulate one experiment under Ho

Assume the Null Hpothesis is true, then the rat is just guessing, with a 50-50 chance of getting the correct answer in each trial. If we code a correct guess as "1" and a wrong guess as "0", how would you simulate a block of 50 trials? [1]

[1] Recall what you learned in lecture 20 about drawing random numbers in R.

```
block <- sample(c(0, 1), 50, replace = TRUE)
block
```

```
##  [1] 0 1 1 1 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 1
## [21] 0 0 1 0 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1
## [41] 0 1 0 0 1 0 0 1 0 0
```

Out of the 50 trials, how many did the rat guess correctly?

```
correct <- sum(block)
```

Simulate 10000 experiments under Ho

Now is where the simulation-based magic happens. Let's say we have not just 1 rat that behaves as if Ho is true, but ten thousand rats. How would you simulate that?[2]

[2] Recall what you learned about loops in R.

First, we define a vector that will hold the number of correct answers per experiment.

```
number_correct <- vector(mode = "numeric")
```

Next, we loop over a counter that goes from 1 to 10000. In each loop, we do a trial just like the one we did before and store the result in `number_correct`

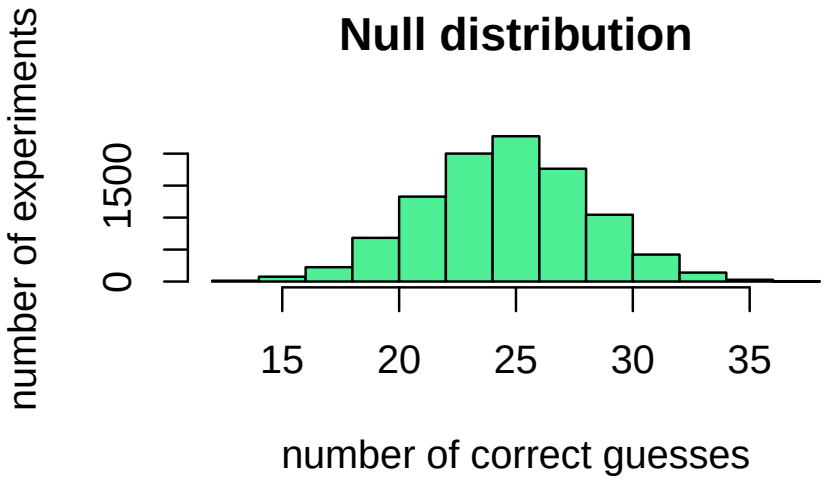
```
nexp <- 10000
for (i in 1:nexp) {
    block <- sample(c(0, 1), 50, replace = TRUE)
    correct <- sum(block)
    number_correct = c(number_correct, correct)
}
```

Visualise simulation results

We have now run the experiment 10000 times assuming H0 is true, and kept track of the number of correct guesses for each experiment. How can we best visualise our results? For instance, by drawing a histogram. Before we do that, answer the following questions for yourself:

- What would you expect the histogram to look like?
- Where would you expect the peak to be?
- Do you remember the name of what this histogram shows you?

```
hist(number_correct, col = "seagreen2", main = "Null distribution",
    xlab = "number of correct guesses", ylab = "number of experiments")
```



The histogram shows the "Null distribution", i.e. the distribution of outcomes, assuming the Null Hypothesis is correct.

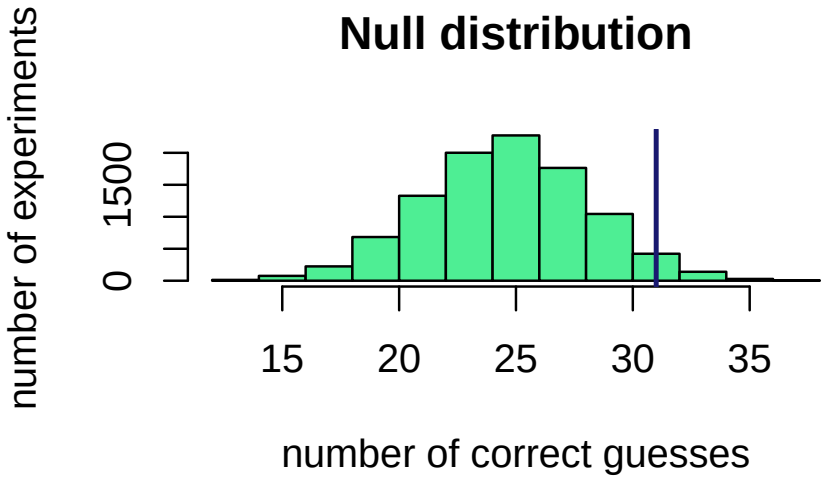
How likely is it to see my experimental results if Ho is true?

As in any hypothesis test, we ask ourselves: "How likely are we to see results as extreme (or even more extreme) as in our experiments if Ho is true?"

First, let's visualise our experimental result (31 out of 50 correct) by adding it as a line to the histogram. This can be done using the abline() function[3], which draws a straight line at a given set of coordinates. What do you see?

[3] Haven't encountered this function before? Remember, you can get more information about any function in R by using help().

```
hist(number_correct, col = "seagreen2", main = "Null distribution",
    xlab = "number of correct guesses", ylab = "number of experiments")
abline(v = 31, col = "midnightblue", lwd = 2)
```



Most of the experiments in the Null distribution are to the left of the blue line (i.e. less extreme than the results we found). But sometimes, even a rat that is guessing gets 31 or more trials correct. But how often exactly? Well, we can just count!

```
sum(number_correct >= 31)
```

```
## [1] 595
```

But really, what we want is not the absolute number, but the proportion of trials under Ho that gave 31 or more correct. Why? Because *that is our p-value*.

```
p = sum(number_correct >= 31)/nexp
```

Interpreting the p-value and coming to a conclusion

- From this p-value, what do you conclude?
- If two of you both run the simulation, do you get the same p value?[4]
- Is ten thousand trials enough? Should there be more? What would you need to change in the code?

[4] Think back to what you learned in lecture 20.1/2 about the exactness of randomisation-based solution.

Bonus: No-loops solution!

Reading the last section, you may have tried to set `nexp` up to one hundred thousand or even a million. You will have found that the resulting p value is more precise, but also that simulations take longer to run. This is because loops are quite time-consuming. Is there a way of achieving what we did without loops?

Yes, there is! If you think about it, we pick between 0 and 1 a total of 50 times per experiment, and a total of `nexp` experiment. So, let's just toss that coin $50 \times nexp$ times to begin with. All we have to do then is be clever about how to sum the number of correct guesses per trial.

```
all_experiments <- sample(c(0, 1), 50 * nexp,
    replace = TRUE)
all_experiments <- matrix(all_experiments, ncol = 50)
number_correct <- rowSums(all_experiments)
p = sum(number_correct >= 31)/nexp
p
```

```
## [1] 0.0608
```

Do men and women have different forearm lengths?

Acknowledgments