Problem Set 2.1: Notes

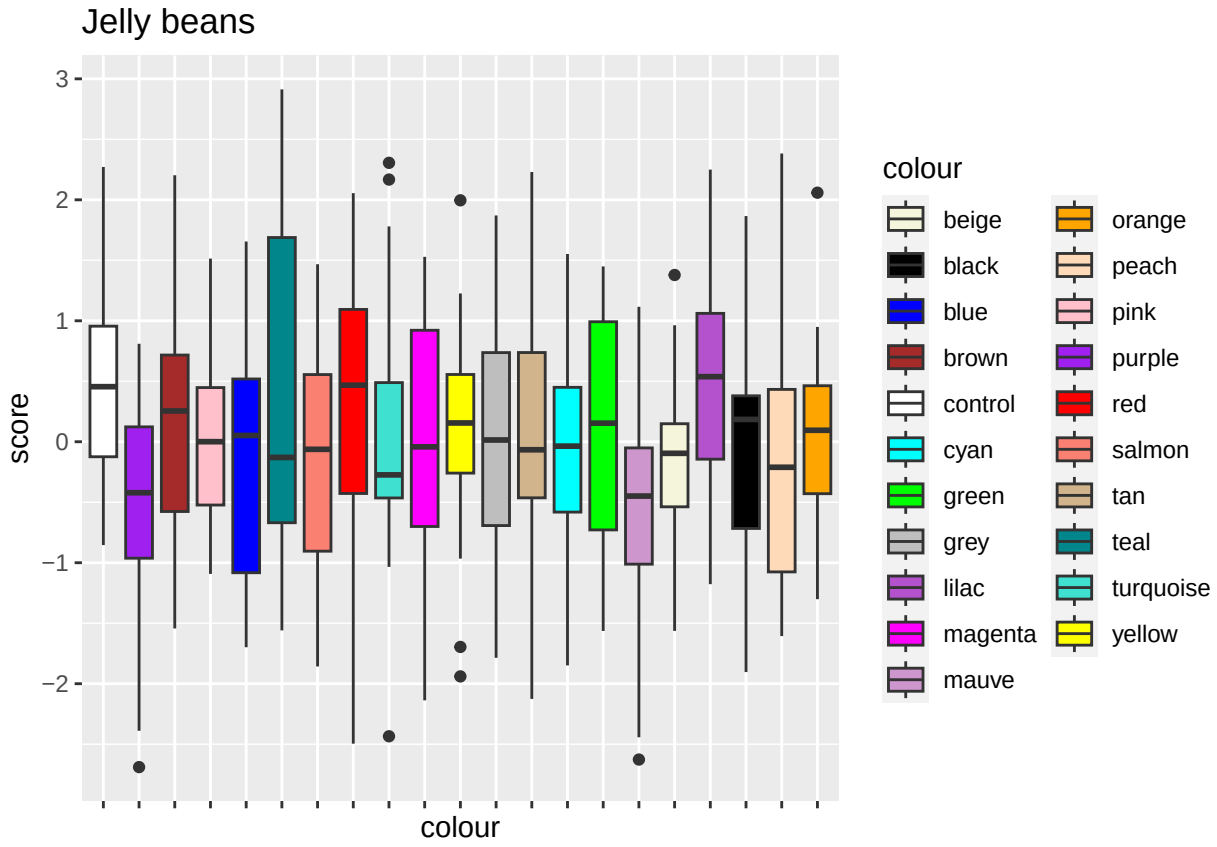
ADS2

Semester 2 2023/24

The problem with jelly beans

Reminder, we were looking at the "jelly bean experiment" from this web comic: https://xkcd.com/882/

I am going to show the plot again, because I like it so much



Madness of multiple t-tests

If scientists compare each of the 20 colours to the control group, that's 20 t-tests. The probability of getting a false-positive result (if they choose $\alpha = 0.05$ for each t-test) is

```
1-(0.95)^(20)
```

```
## [1] 0.6415141
```

In other words, a 64% chance that there will be at least one false positive result. From just eyeballing the

plot, it seemed to me that even already the first comparison (control vs purple) looked iffy, so I went ahead and did a t-test for that:

```
control <- jellybeans[jellybeans$colour=="control","score"]
purple <- jellybeans[jellybeans$colour=="purple","score"]
t.test(control, purple)
```

```
##
##  Welch Two Sample t-test
##
## data:  control and purple
## t = 3.6464, df = 37.215, p-value = 0.0008089
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.4614498 1.6151098
## sample estimates:
##  mean of x  mean of y
##  0.4350580 -0.6032218
```

And indeed, the t-test gives a positive result. (I know it is a *false* positive, because I generated the dataset myself by sampling from the same distribution for every colour, so in theory, there really is no difference)

Can we look at variance instead?

If we decide to look at the variance instead, the main question is: can differences between data points be explained by them belonging to different groups? We can draw random pairs that belong either to the same or to different groups and record their differences. Similar to the practical, we can then take the average difference within groups and the average difference between groups for our particular dataset. Then run the randomisation test to create a null distribution and compare our result to the null distribution in order to get a p value.

We did not ask you to encode this in detail, because we are more interested in the general idea. But if you *are* interested in the nitty-gritty detail, here is one way of doing it:

First, we want to create a function that draws random pairs from a dataset, computes their difference, keeps a list of differences between points in the same group and differences between points in different groups, takes the mean of each of those lists and compares the means.

One simple way to go about this is as follows:

permutation test

- Draw 2 random numbers between 1 and the number of rows in the dataset without replacement (so they will be different)
- Those are the row numbers of the two sample points to compare. Compute the absolute difference between them
- Check whether they belong to the same colour or to different colours. If they belong to the same colour, add them to the "same group" list, otherwise add them to the "different groups" list.

This is fine in principle, but a bit impractical if you have many groups. In our case, we have 21 groups, so if we sample randomly 1000 times, we can expect around 950 pairs from different groups, but only 50 pairs from the same group. So in order to get a decent sample size for the "same group" list, you would have to sample many times.

What we will do instead is this: Draw one data point, and then draw another one from the same group and another one from a different group. If you do this 1000 times, you will get exactly 1000 data point in each of the lists - easy.

We will do this by doing the sampling in two steps: First, select the two colours to be sampled from. Second, select two different points from one colour, and a third one from the other colour.

```r
# create compare_pairs function.
# Inputs: name of dataset, number of draws
# draws random pairs of sample points that are either in the same group or in different groups
# computes their absolute distance
# computes the mean distances in the same group and the mean of distances in different groups
# returns the difference between those mean distances

compare_pairs <- function(dataset, ndraws) {
  same_group <- {}
  between_group <- {}
  for (i in 1:ndraws) {
    # determine condition (colour) to sample from
    list_colours <- unique(jellybeans$colour)
    colours <- sample(list_colours, 2, replace=FALSE)
    # Make spearate lists for each group
    # (not strictly necessary, but helpful)
    firstgroup <- jellybeans[jellybeans$colour == colours[1], "score"]
    secondgroup <- jellybeans[jellybeans$colour == colours[1], "score"]
    # draw samples
    s1s2 <- sample(firstgroup, 2, replace=FALSE)
    s3 <- sample(secondgroup, 1)
    # compute same group and between group differences, add to list
    same_group <- c(same_group, abs(s1s2[1] - s1s2[2]))
    between_group <- c(between_group, abs(s1s2[2] - s3))
  }
  # compute means of same-group and between-group differences
  mean_same <- mean(same_group)
  mean_between <- mean(between_group)
  # compute absolute difference between those means
  diffmeans = abs(mean_same-mean_between)
  return(diffmeans)
}
```

Let's try it! (The number of draws necessary to get a good result is not immediately obvious. We'll use 10000 here, but there is obviously a trade-off to think about between how exact your result is and how long it takes to run.)

```r
our_experiment <- compare_pairs(jellybeans, 10000)
our_experiment
```

```
## [1] 0.0689206
```

Now, to get a p value, we just need to compare the two groups ("same" and "different").

There are different ways to do this. In the practical, we used a Wilcoxon rank-sum test and of course you can do that here as well.

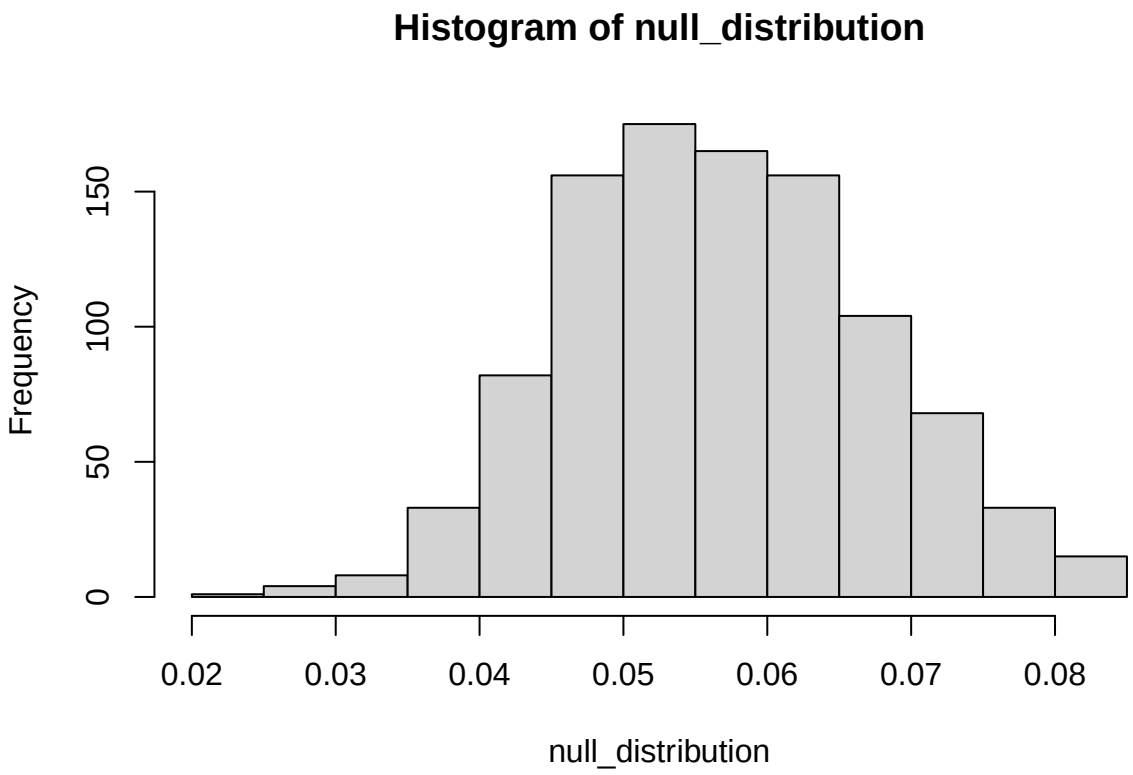
Alternatively, we can do a randomisation test like we learned in semester 1. We will do this now. The idea is, if H0 is true, then the labels (colours), are meaningless, so we can randomly re-assign data points to groups. We will do this 1000 times (this takes several minutes).

```r
# create empty set for null distribution
null_distribution <- {}
for (j in 1:1000) {
  # make new dataframe called random_experiment
  # and shuffle colour column
  random_experiment <- jellybeans
```

```
  random_experiment$colour <- sample(jellybeans$colour,
                                     nrow(jellybeans), replace=FALSE)
  # use compare_pairs function on random_experiment
  random_meandiff <- compare_pairs(random_experiment, 10000)
  # add the result to the Null distribution
  null_distribution <- c(null_distribution, random_meandiff)
}
```

OK, so this gives us a Null Distribution, which we can look at

```
hist(null_distribution)
```



And now, to compute the p-value, we ask ourselves: If H0 is true, how many times do we get a result as or more extreme as we saw in our experiment? (In other words, what proportion of values in the Null Distribution is greater than or equal to the number we found in our original dataset?)

```
p_value <- sum(null_distribution >= our_experiment) / length(null_distribution)
p_value
```

```
## [1] 0.139
```

What do you conclude?

There is a difference, what now?

If we find a significant p value in an ANOVA, then we can conclude that there is a difference between at least two groups, but we don't know where exactly the difference is.

To find this out, we do a *post-hoc test*. This is essentially a bunch of t-tests, but we need to adjust the alpha level for each individual test, so that the overall alpha level is 0.05. Remember for 6 t-tests, if H0 is true, the overall false positive probability is:

$$P(falsepositive) = 1 - (1 - \alpha)^6$$

We need:

$$P(falsepositive) = 0.05$$

This means we can get the individual α values we need by re-arranging

$$0.05 = 1(1 - \alpha)^6$$

$$(1 - \alpha)^6 = 1 - 0.05$$

$$(1 - \alpha)^6 = 0.05$$

$$1 - \alpha = 0.95^{\frac{1}{6}}$$

$$\alpha = 1 - 0.95^{\frac{1}{6}}$$

We can plug the numbers into R to get

```
alpha = 1 - 0.95^(1/6)
alpha
```

```
## [1] 0.008512445
```

Last update by DJ MacGregor in 2024