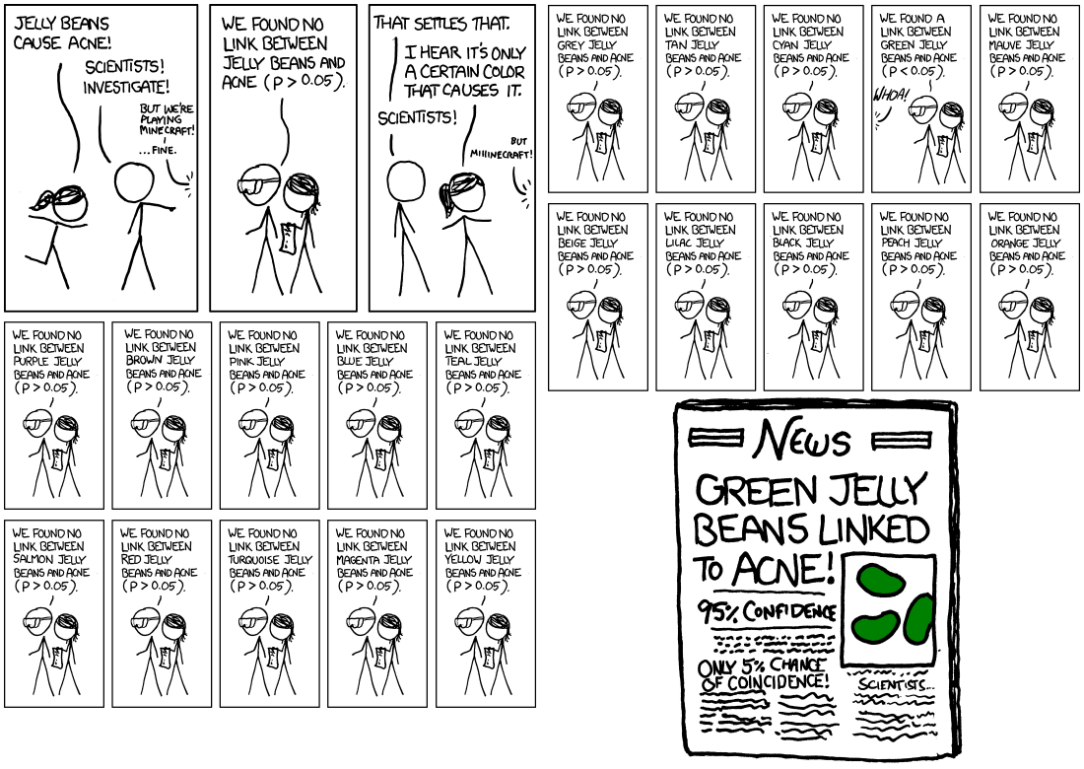
Quantitative Understanding in Biology
1.6 Multiple Hypothesis Testing and Non-parametric Tests

Jason Banfelder

September 23rd, 2021



'So, uh, we did the green study again and got no link. It was probably a–'
'RESEARCH CONFLICTED ON GREEN JELLY BEAN/ACNE LINK; MORE STUDY RECOMMENDED!'

http://xkcd.com/882/

1 Multiple Hypothesis Testing

So far, we have considered one statistical test at a time. We know that we can control the rate of Type I errors in these tests by setting α appropriately. In many studies several hypotheses will be tested; for example, we may wish to compare means for several groups (related or independent). We may be testing a drug on several different cell lines, or we may be measuring a response at several different time points after application of a treatment. The proper way to analyze data from studies such as these varies depending on the specifics of the experimental design, and we won't be able to go into detail for all them. There is, however, a common theme that needs to be considered: that of multiple hypothesis testing.

The challenge in dealing with multiple hypothesis testing is controlling the Type I error rate across the whole study. If a study involves testing 20 hypotheses, and each has a 5% chance of a Type I error, then the probability of at least one false conclusion is:

```
1 - dbinom(0, 20, 0.05)
```

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

In other words, there is a 64% chance that at least one conclusion in our study is wrong. If we want to control the error rate for the study as a whole, we need to adjust α downwards in an appropriate manner. In practice, there are several approaches that are in common use; we'll look at two.

2 The Bonferroni Correction

The simplest and most conservative approach to controlling a study-wide error rate is the Bonferroni correction. Given a desired study-wide error rate, α, you compute a per-test cutoff, α^*, as follows...

$$\alpha^* = \frac{\alpha}{n} \tag{1}$$

...where n is the number of hypothesis tests in your study. In our example above, we compute $\alpha^* = 0.0025$. Using this new-per test cutoff, we can estimate the probability of one or more false conclusions...

```
1 - dbinom(0, 20, 0.05 / 20)
```

```
## [1] 0.04883012
```

...which is quite close to what we wanted.

In practice, one usually doesn't adjust α, but rather we 'correct' the p-value. For the Bonferroni correction, we multiply the raw p-value by n to compute the corrected p-value, and compare that to our desired study-wide α of 0.05. While this is a little easier in terms of book-keeping, it should be kept in mind that a 'corrected' p-value is not a probability of any particular scenario we've tested. In fact, corrected p-values can be larger than unity (although they are usually reported as 1 in this case).

The Bonferroni correction is the most conservative correction used in multiple hypothesis testing. When the number of hypotheses is small, this is probably an appropriate correction to use.

One of the difficulties in critically evaluating scientific literature is that publications are biased toward reporting statistically significant results. When you see a paper that reports a p-value of 0.02 for a particular test, you have no way of knowing how many other hypotheses have been tested and not reported by the authors.

There can also be a problem when you are 'just looking' at some data you have collected to decide how to analyze it. You are implicitly performing many tests on the data, and selecting only those for which the numbers look hopeful to compute a p-value for. In principle, you should be using some kind of multiple hypothesis control in this case.

Ideally, you would design your experiments and your analyses before collecting any data, and all results, statistically significant or not, would be published. As this is not likely or practical in today's scientific and publishing landscape, it is important to recognize that reported results are probably less certain than they might appear.

3 The Benjamini-Hochberg Correction for Controlling False Discovery Rate

The advent of high-throughput biological techniques has resulted in a renewed interest in multiple hypothesis testing. New techniques such as microarray and high-throughput sequencing experiments allow for many thousands of data points to be collected in a single experimental protocol; as a result, it is not uncommon to test tens of thousands of hypotheses in a single analysis. In such cases, many practitioners find that the Bonferroni correction is too conservative.

The most common alternative in use today is the Benjamini-Hochberg correction. It works as follows:

1. Order all p-values from smallest to largest, and assign a rank to each one.
2. Correct each p-value by multiplying it by $\frac{n}{\text{rank}}$. This leaves the smallest with the

same adjusted p-value as would have been obtained using the Bonferroni correction and the largest p-value uncorrected.

3. Compare the corrected p-values to your pre-determined α, stopping at the first case where the correct p-value is not less than α; all subsequent tests are deemed to be non-significant.

Note that the Benjamini-Hochberg correction seeks to control the "False Discovery Rate", not the "Family-wise Error Rate". This means that we expect some fraction of the significant result to contain false positives. It is the least conservative correction for multiple hypothesis testing in common use today (short of no correction at all). As such, it is usually applicable to screening studies, where we are trying to identify an enriched set of target genes or compounds for further study, and is usually not the basis on which final scientific conclusions are based.

4 Doing Multiple Hypothesis Testing: A Worked Example

The Bonferroni and Benjamini-Hochberg p-value corrections (as well as some others) are available in R. The relevant function is `p.adjust`. You need to provide a method argument indicating which correction you want to use; be sure to specify `method = 'BH'` (and not `method = 'hochberg'`) for the Benjamini-Hochberg correction discussed here. Further details are available in the help page for the `p.adjust` function.

Here we will work an example that demonstrates how the Benjamini-Hochberg correction controls for the False Discovery Rate, and why it is often the preferred p-value correction method for screens; this should help make clear what exactly the FDR is.

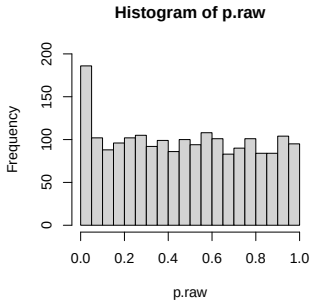
Suppose we are screening a library of 2,000 compounds for a differential effect. For each compound, we'll assay six control replicates, and six treatment replicates, and then run a t-test to compare the means between the two groups. To make things easy on us (at least at first), we'll assume that, for those compounds that do have an effect, the effect is strong (d = 3). You should be able to do a power calculation that shows that there is a very high chance ($> 99\%$) that for compounds that have an effect, a basic t-test will be statistically significant. Also, while still in the role of the omniscient, we'll say that the first 100 compounds are efficacious, and the remaining 1,900 are not. Of course, in "real life" your efficacious compounds would be scattered around your library, and you wouldn't know *a priori* how many truly active compounds there are. But, as usual, when exploring new methods, it helps to invent a 'truth' and see how well the method deals with it.

The following code generates data according to this model:


```
true.count <- 100
compound.count <- 2000
n <- 6
effect.size <- 3.0
x <- list()
y <- list()
for (idx in 1:compound.count) {
  x[[idx]] <- rnorm(n, mean = 0)
  y[[idx]] <- rnorm(n, mean = ifelse(idx <= true.count, effect.size, 0.0))
}
```

Next, we'll compute the raw (unadjusted) p-values for each of our 2,000 cases, and make a histogram of them.

```
  p.raw <- numeric(length = compound.count)
  for (idx in 1:compound.count) {
    p.raw[idx] <- t.test(x[[idx]], y[[idx]])$p.value
  }
  hist(p.raw,
       breaks = seq(from = 0, to = 1, by = 0.05),
       ylim = c(0, 200))
```



We've prepared the histogram with bin widths of 0.05, so the left-most bar contains all of the statistically significant results. The histogram indicates that there are just shy of 200 significant results, which is just about what you'd expect: there are 100 'real' results, and our power calculation tells us that we're going to find nearly all of them; plus there should be a Type I error rate of 5%, so we expect $1900 \cdot 0.05 = 95$ false positives.

One important observation to make here is that the distribution of p-values for the true negative cases should follow the uniform distribution (i.e., the histogram should be flat), as we see here. This is generally true, and can be a very useful quality-control metric in studies where you know you have many, many true negatives. It is particularly useful in

GWAS studies, where the entire genome is interrogated for effect on a phenotype, and you (more or less) know that only a few regions actually affect the phenotype in a meaningful way.

If we perform a Bonferroni correction on our p-values, we see that only 14 values pass the correction.

```
  sum(p.adjust(p.raw, method = 'bonferroni') < 0.05)
```

```
## [1] 14
```

You should understand that we are 95% sure that all 14 of these are true positives; i.e., there is only a 5% chance that there is one or more mistakes in our list. Since we know that the first 100 compounds are the efficacious ones, we can check this...

```
  which(p.adjust(p.raw, method = 'bonferroni') < 0.05)
```

```
##  [1]  3 12 16 35 42 49 52 58 59 69 83 86 93 99
```

...and confirm that, indeed, we have not made any errors.

Unfortunately, we've lost many other hits because of the strictness of the Bonferroni correction. And that was with a strong signal! Let's see how the Benjamini-Hochberg correction fares[1].

```
  sum(p.adjust(p.raw, method = 'BH') < 0.05)
```

```
## [1] 81
```

```
  which(p.adjust(p.raw, method = 'BH') < 0.05)
```

```
##  [1]    3    4    5    6    7    8    9   10   11   12   13   15   16
## [14]   18   19   20   21   22   23   27   29   31   32   33   34   35
## [27]   37   38   39   40   41   42   43   44   45   46   47   48   49
## [40]   50   51   52   53   54   55   56   57   58   59   63   64   67
## [53]   68   69   73   75   76   77   78   80   81   82   83   85   86
## [66]   87   88   89   90   91   92   93   94   95   97   98   99  189
## [79]  207  971 1820
```

Here we count 81 hits. Looking at the indices of the hits, we see that they are all true positives, except for the last four. This gives us a False Discovery Rate of $4/81 = 0.049$, which is pretty close to the expected FDR of 5%.

[1] R's algorithm for correcting the p-value by FDR is different than described here, and is formulated so that simple code like this yields the correct list of hits. You can see how R does it by looking at the source code (just type the function name, `p.adjust`, without the parentheses). If you're interested, convince yourself that the method described above and R's implementation yield the same set of hits. One of the advantages of using open source software like R is that full details of all methods are available for inspection.

From the example, you can see why control of FDR is often used in large-scale screening studies. You probably wouldn't want to publish your list of hits from just a Benjamini-Hochberg corrected screen as a definitive list (although too many people do!), but it can be an efficient way to generate high quality leads to be validated by more rigorous (and probably expensive) methods.

It can be instructive to repeat this exercise using smaller effect sizes; try it with an effect size of 1.0, and 2.0; don't forget to do the power calculations for these effect sizes as well!

5 The Randomization Test: An Example of Testing By Simulation

In a previous session, we saw how to compare two means using the t-test. This test is based on a model in which the data from the two populations are normally distributed and have the same[2] SD. An alternative method for comparing two means, which does not make these assumptions, is called the randomization test. In practice, the randomization test is used rarely, if ever. However, it is interesting because it works without the need for any complex modeling or assumptions. Additionally, the method forms the basis of a non-parametric test for comparing two means, which we will cover shortly.

We begin with two sets of observations, and their means:

$$\begin{aligned} x_1, x_2, x_3, x_4, x_5 \qquad & \overline{x} \\ y_1, y_2, y_3, y_4, y_5 \qquad & \overline{y} \end{aligned} \tag{2}$$

The difference between the two observed means is

$$\Delta = \overline{y} - \overline{x} \tag{3}$$

As with the t-test, we wish to ascertain whether this observed difference is statistically significant, or if it could be due to chance. Our null hypothesis is that the two sets of observations are samples from the same distribution. Interestingly, for the randomization test we do not need to assume anything about this hypothesized distribution.

Now, if the null hypothesis were true, then any of the values we observed would be just as likely to appear in the first set as in the second. In other words, any rearrangement

[2] By default, R's `t.test` function includes a correction for this assumption; many other programs don't do this unless you ask.

or shuffling of the values we observed (keeping the count of values in each group the same) is just as likely to have been observed as the arrangement we did in fact observe. We can therefore enumerate every possible rearrangement of the values we observed, and compute a Δ for each one. We then have a histogram of Δs that can serve as an estimate for the probability distribution function of Δ. Using this approximate distribution, we can compute the probability of observing given differences in means from two random samples from our hypothesized distribution. We can then compute what proportion of those Δs is equal to or larger in magnitude than the one we observed. This is the p-value corresponding to our null hypothesis. You can look at the range of Δs, and compute a CI of your choosing.

Of course, this p-value is only an estimate. Interestingly, it is a proportion, so, if you are motivated, you can compute a CI for the p-value using techniques we learned earlier.

As mentioned above, the randomization test is rarely, if ever, used in practice. It involves a good deal of bookkeeping to elencate all of the possible rearrangements of the observed values. For all but the most trivial cases, you would need a computer. Even so, with more than a moderate count of observations in each group, the resultant combinatorial explosion would be beyond the capacity of even the most powerful computers. In such cases, sampling a reasonably large number of rearrangements would allow you to develop an estimate of the distribution of Δ, and would allow you to approximate a p-value. The exhaustive procedure is known as the (ostensibly oxymoronic) exact randomization test!

Again, the randomization test is hardly ever used in practice; most sane people would use the t-test if they were comfortable with its assumptions regarding normality and equivalent SDs. That said, if you are comfortable with the idea behind the randomization test, then you have a good understanding of what a p-value is.

6 The Wilcoxon Rank-Sum Test

The Wilcoxon Rank-Sum test is similar in spirit to the randomization test, and is in fairly common use. It is a non-parametric test that seeks to answer a similar question to the t-test: we again have two sets of observations, and we wish to ascertain whether they come from the same distribution. We are not comfortable with the assumptions of the t-test, and choose a non-parametric method.

Again, we begin with two sets of observations (same as above). We begin our analysis by ordering the values from smallest to largest, and associating a rank with each observation (in the event of a tie, use the average of the ranks to be assigned). Our order might be...

x_4	y_1	y_3	x_2	y_5	y_4	y_2	x_1	x_5	x_3
1	2	3	4	5	6	7	8	9	10

. . . and finally a Δ for the difference between the rank sums would be. . .

$$\Delta = 23 - 32 = -9 \tag{4}$$

Our reasoning from this point on is analogous to that of the randomization test. If the samples were from the same underlying distribution, then any rearrangement or shuffling of the data would be just as likely as the arrangement we observed. We can therefore develop a distribution of Δs, and estimate a p-value that indicates how likely we are to see a Δ as large in magnitude as the one we actually observed.

Note that we never used the actual observed values, just their order. As you might imagine, the Wilcoxon Rank-Sum test is quite robust to outliers; it doesn't matter if the largest value is 100 or 10,000,000; the result would be exactly the same.

The relevant function in R is `wilcox.test`; see the help for details.

Note that the test outlined above is sometimes referred to as the "Mann-Whitney test". To be very precise, we should call it the "two sample Wilcoxon rank sum test". The 'one sample' version is a non parametric test used for paired data, and is available in R using the same `wilcox.test` function, but with the `paired = TRUE` option.

7 Publication Quality Figures with R

In addition to being a powerful data analysis platform, R is a great tool for preparing publication quality figures. The golden rule for technical illustration is to avoid using any bitmap formats. Sticking with vector based formats allows you to edit your figures with exquisite precision (Affinity Designer or Adobe Illustrator are recommended for this; PowerPoint is specifically recommended against), and means that you can shrink or enlarge them without loss of clarity. This last point is especially important when preparing posters; you would otherwise have to save your figures with ridiculously high resolution to avoid seeing pixelation and ugly anti-aliasing effects.

To save a figure in R as a PDF (a handy vector format that nearly everyone can read and is compatible with Affinity Designer and Adobe Illustrator), you begin by opening a PDF file as a graphics device (the file is created in R's current directory):

```
pdf(file = 'x.pdf')
```

Next, you issue your normal plotting commands. Instead of appearing on the screen, these commands will be sent to the current graphics device, which is your PDF file.


```
x <- rnorm(100)
hist(x, probability = TRUE)
rug(x)
```

If you make multiple plots, a multi-page PDF will be created. Making plots in a loop will create a "book" of all your plots, with one action.

To close the device, you issue the `dev.off()` command. Subsequent plotting commands will appear on the screen as usual.

```
dev.off()
```

R Studio also allows you to export plots (one at a time) from the Plots tab.

You can also have multiple graphics devices open at once. This allows you to have, say, two windows on the screen, or a screen window and a PDF file open at the same time. Use the `x11()`, `windows()`, and `quartz()` functions to open new screen graphics devices on Linux, Windows, and Mac, respectively.

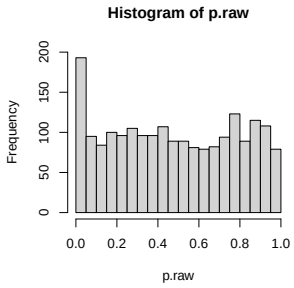
Plotting commands always go to the current device, which you can change with the `dev.set` function. For help on this function and its friends, see the R help: `?dev.set`

For simplicity, we are limiting ourselves to R's so-called "base graphics" for most of this course. There is a very widely used package called `ggplot2` that creates excellent plots and is very powerful. Any serious R user should learn `ggplot2` and other allied packages in the tidyverse. For inspiration, we'll demonstrate `ggplot2` in action at the end of this course.

8 Exercise

n.b. Due to the vagaries of random sampling, the results you obtain may vary slightly from those shown here, but they should agree qualitatively.

1. Redo the simulation in Section 4 using an effect size of 2.5 instead of 3.0, and plot a histogram of the raw p-values.



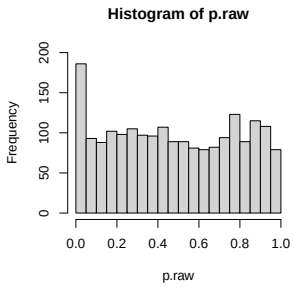
2. When a Bonferroni correction is applied to your simulated data using $\alpha = 0.05$, how many true positives do you recover? How many false positives do you find? How many true negatives and false negatives do you have?

```
##   TP   FP   FN   TN
##    1    0   99 1900
```

3. When a Benjamini-Hochberg correction is applied to your simulated data using $\alpha = 0.05$, how many true positives do you recover? How many false positives do you find? How many true negatives and false negatives do you have?

```
##   TP   FP   FN   TN
##   34    0   66 1900
```

4. Repeat all of the above, but now with $d = 2.0$




```
## [1] "Bonferroni..."
##   TP   FP   FN   TN
##    1    0   99 1900
```

```
## [1] "Benjamini-Hochberg..."
##   TP   FP   FN   TN
##    1    0   99 1900
```

As demonstrated, when the effect size is small, the correction method won't necessarily rescue your study.

