Introductions (if anyone is new)

- Name
- Year
- Make sure to sign in on the google form (linked here)

Counting, ranks, and birthdays

1. Suppose we have samples from two groups $Y_{1,1}, \dots, Y_{1,n}$ and $Y_{2,1}, \dots, Y_{2,m}$ of sizes n and m respectively where $Y_{i,j}$ all have the same distribution, and assume we are measuring a continuous variable (so every $Y_{i,j}$ is unique). Rank the whole data set and keep just the ranks for the first group: $Z_{1,1}, \dots, Z_{1,n}$. Let the set of ranks for the first group be $S_1 : \{Z_{1,1}, \dots, Z_{1,n}\}$. How many possible sets of ranks are there for the first group? How many for the second group?

$$\binom{n+m}{n}, \binom{n+m}{m}$$

2. Suppose we rerun this experiment r times. What is the probability that at least two of the reruns will give you the same set of ranks for the first group.

Let $t = \binom{n+m}{n}$. Then, because each set of ranks is equally likely, the probability is

$$1 - \left[\frac{t}{t} \cdot \frac{t-1}{t} \cdot \frac{t-2}{t} \cdot \dots \cdot \frac{t-r+1}{t}\right]$$

3. Use an R function to find this probability for the case $n = 20$, $m = 10$, and $r = 1000$.

```r
n = 20
m = 10
r = 1000
t = choose(m+n, n)

# Built-in
pbirthday(r, t)
```

```
## [1] 0.0164878
```

```r
# By hand
1-prod((t - (0:(r-1)))/t)
```

```
## [1] 0.0164878
```

Variance by decomposition

Let $X \sim \text{Bin}(n, p)$ and $Y \sim \text{Bin}(m, p)$. Let $X + Y = r$.

1. Find the variance of $X|r$ by using the variance of a known distribution (See 3.9.2 in the Stat 110 book for a hint).

$X|r \sim \text{HGeom}(n, m, r)$, so by the variance of the hypergeometric we have:

$$\text{Var}(X|r) = \frac{n+m-r}{n+m-1} \cdot \frac{nr}{n+m} \cdot \frac{m}{n+m}$$

2. Find the variance of $X|r$ by using the fact that $\text{Var}(X + Y) = 0$ and treating X and Y as the sum of Bernoulli random variables. Verify that the two answers are the same. (Hint: Once you get to the Bernoulli random variables, think about how knowing the sum is r makes p irrelevant.)

First, note that the variance of a constant is 0, so $\text{Var}(X+Y|X+Y=r)=0$. Each of X and Y can be decomposed into Bernoullis, and each of these have the same variance and covariance by symmetry. Therefore, $0=\text{Var}(X+Y|X+Y=r)=(n+m)\text{Var}(I_i|r)+2\binom{n+m}{2}\text{Cov}(I_i,I_j|r)$ for $i\neq j$. Then, we can solve for the covariance: $\text{Cov}(I_i,I_j|r)=-\text{Var}(I_i|r)/(n+m-1)$. Conditioning on r, $P(I_i=1|r)=\frac{r}{n+m}$, so $\text{Var}(I_i|r)=\frac{r}{n+m}\cdot\frac{n+m-r}{n+m}$ and $\text{Cov}(I_i,I_j|r)=-\frac{r}{n+m}\cdot\frac{n+m-r}{(n+m-1)(n+m)}$. Then, building up X again,

$$\begin{aligned}\text{Var}(X|r)&=\text{Var}(\sum_{i=1}^n I_i)\\&=n\cdot\frac{r}{n+m}\cdot\frac{n+m-r}{n+m}-2\binom{n}{2}\frac{r}{n+m}\cdot\frac{n+m-r}{(n+m-1)(n+m)}\\&=\frac{nr(n+m-r)}{(n+m)^2}\left[1-\frac{n-1}{n+m-1}\right]\\&=\frac{nmr(n+m-r)}{(n+m)^2(n+m-1)}\end{aligned}$$

Everything everywhere all at once (all two-sample continuous comparisons)

Let $X_1,\dots X_n\sim\text{Exp}(1/\mu_1)$ and $X_1,\dots X_m\sim\text{Exp}(1/\mu_2)$ with $n=5$, $m=15$, $\mu_1=5$, and $\mu_2=3$.

1. Name three tests we've learned so far that would not be applicable here.

ANOVA (only two groups), paired t-test (different number of observations in each group), proportion test (not proportions)

2. Fill in the following code to calculate a one sample t-based test of $H_0:\mu_1=\mu_2$, a log-transformed test for $M_1=M_2$, a rank-based test for $M_1=M_2$, and a permutation test for exchangability between the groups. Find the power of each test.

```
set.seed(139)
mu_1=5
mu_2=3
n=5
m=15

nsims = 5000
t_test_out = vector(length = nsims)
transformed_t_out = vector(length = nsims)
rank_out = vector(length = nsims)
perm_test_out = vector(length = nsims)

nboot = 250

for (i in 1:nsims) {
  x = rexp(n, 1/mu_1)
  y = rexp(m, 1/mu_2)
  t_test_out[i] = t.test(x, y)$p.value
  transformed_t_out[i] = t.test(log(x), log(y))$p.value
  rank_out[i] = wilcox.test(x, y)$p.value
  df = cbind(c(x,y), c(rep(0, n), rep(1, m)))
  boot_diff = vector(length = nboot)
  for (j in 1:nboot) {
    df_tmp = cbind(df[,1], sample(df[,2], n+m))
    boot_diff[j] = mean(df_tmp[df_tmp[,2]==0,1]) - mean(df_tmp[df_tmp[,2]==1,1])
```

```r
  }
  perm_test_out[i] = mean(abs(boot_diff) >= abs(mean(x) - mean(y)))
}

output = data.frame(cbind("test" = c("t test", "log t test", "rank test", "perm test"),
               "correct" = c(mean(t_test_out <= 0.05), mean(transformed_t_out <= 0.05),
                             mean(rank_out <= 0.05), mean(perm_test_out <= 0.05)),
               "wrong" = c(mean(t_test_out > 0.05), mean(transformed_t_out > 0.05),
                             mean(rank_out > 0.05), mean(perm_test_out > 0.05))))

print(output)
```

```
##         test correct  wrong
## 1     t test  0.0506 0.9494
## 2 log t test  0.1596 0.8404
## 3  rank test  0.1152 0.8848
## 4  perm test  0.2196 0.7804
```

Note what an extreme violation of the t-test assumptions we needed to break the t-test.

3. Make a short change that would evaluate the false positive rate of each.

Change μ_1 to equal μ_2, and change "correct" and "wrong."

4. What assumptions do we need for each test and what hypotheses are we testing?

- t-test
  - Assumptions: independence and normality
  - Hypotheses: H_0 : Means are equal; H_a: Means are not equal.
- Log-transformed t-test
  - Assumptions: independence and symmetry once transformed
  - Hypotheses: H_0 : Ratio of medians is 1; H_a : Ratio of medians isn't 1.
- Rank-based test
  - Assumptions: independence; $n_1, n_2 \geq 10$
  - Hypotheses: H_0 : True average quantile (median) in the two groups within the population are the same, H_a: There is an association between group status and the average quantile (median) of outcomes in the population
- Permutation test
  - Assumptions: independence
  - Hypotheses: H_0 : Distribution of outcomes is not related to group status. H_a : It is related.

5. Fill in the following code to calculate a t-based confidence interval for the difference in means, a t-based confidence interval for the ratio of medians, a studentized bootstrap interval for the difference in means, a percentile bootstrap interval for the difference in means, and a reversed percentile bootstrap interval for the difference in means. Find the coverage probability for each test.

There is a bug in this code; refer to the 2023 version for the correct simulation.

```r
t_cap = vector(length = nsims)
t_length = vector(length = nsims)
t_med_cap = vector(length = nsims)
t_med_length = vector(length = nsims)
t_boot_cap = vector(length = nsims)
t_boot_length = vector(length = nsims)
perc_boot_cap = vector(length = nsims)
perc_boot_length = vector(length = nsims)
rev_boot_cap = vector(length = nsims)
rev_boot_length = vector(length = nsims)
```

```r
df = min(n-1, m-1)

for (i in 1:nsims) {
  x = rexp(n, 1/mu_1)
  y = rexp(m, 1/mu_2)

  # T-based interval
  t_out = t.test(x, y)$conf.int
  t_cap[i] = mu_1-mu_2 >= t_out[1] & mu_1-mu_2 <= t_out[2]
  t_length[i] = t_out[2] - t_out[1]

  # Transformed t interval
  t_med_out = t.test(log(x), log(y))$conf.int
  t_med_cap[i] = qexp(0.5, 1/mu_1)/qexp(0.5, 1/mu_2) >= exp(t_med_out[1]) &
    qexp(0.5, 1/mu_1)/qexp(0.5, 1/mu_2) <= exp(t_med_out[2])
  t_med_length[i] = exp(t_med_out[2]) - exp(t_med_out[1])

  # Bootstrap
  boot_diff = vector(length = nboot)
  for (j in 1:nboot) {
    x_star = x[sample(1:n, n, replace = T)]
    y_star = y[sample(1:m, m, replace = T)]
    boot_diff[j] = mean(x_star) - mean(y_star)
  }

  marg = qt(0.975, df) * sqrt(n/(n-1)) * sd(boot_diff)
  lb = mean(x) - marg
  ub = mean(x) + marg

  t_boot_cap[i] = mu_1-mu_2 >= lb & mu_1-mu_2 <= ub
  t_boot_length[i] = ub - lb

  perc_boot_cap[i] = mu_1-mu_2 >= quantile(boot_diff, 0.025) &
    mu_1-mu_2 <= quantile(boot_diff, 0.975)
  perc_boot_length[i] = quantile(boot_diff, 0.975) - quantile(boot_diff, 0.025)

  bounds = mean(x)-mean(y) -
    (quantile(boot_diff, c(0.025, 0.975)) - (mean(x)-mean(y)))

  rev_boot_cap[i] = mu_1-mu_2 >= bounds[2] & mu_1-mu_2 <= bounds[1]
  rev_boot_length[i] = bounds[1] - bounds[2]
}

output = data.frame(cbind("name" = c("t interval", "log t interval", "studentized boot",
                                     "percentile boot", "reverse boot"),
                          "capture proportion" = c(mean(t_cap), mean(t_med_cap),
                             mean(t_boot_cap), mean(perc_boot_cap), mean(rev_boot_cap)),
                          "length" = c(mean(t_length), mean(t_med_length),
                             mean(t_boot_length), mean(perc_boot_length),
                            mean(rev_boot_length))))

print(output)
```

```
##                name capture.proportion           length
## 1       t interval             0.8968 10.8888104626939
## 2   log t interval             0.9442 8.84687934259462
## 3 studentized boot              0.955 12.0079314936131
## 4  percentile boot              0.824 7.32662972950945
## 5     reverse boot              0.828 7.32662972950945
```

6. Based on the results above, which is the best confidence interval to use?

In general, there appears to be a trade-off between capture proportion and confidence interval length. However, the studentized bootstrap has a capture proportion closest to the nominal confidence level, so we should use it.

7. Imagine now that we wanted to construct a confidence interval for μ_1 by using a studentized bootstrap interval. If we knew the data were distributed exponentially, what's one way we could make the confidence interval for μ_1 equally as wide or narrower while keeping the same confidence level?

Make the lower bound 0 if it's ever negative in the confidence interval.