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?

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.

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

# TODO: Calculate probability
```

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).

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.)

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.

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.

```r
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] = # TODO
  transformed_t_out[i] = # TODO
  rank_out[i] = # TODO
  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])
  }
  perm_test_out[i] = # TODO
}

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)
```

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

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

- *t*-test
  - Assumptions:
  - Hypotheses:
- Log-transformed *t*-test
  - Assumptions:
  - Hypotheses:
- Rank-based test
  - Assumptions:
  - Hypotheses:

- Permutation test
  - Assumptions:
  - Hypotheses:

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 and the width of the confidence interval.

```
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)

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_cap[i] = # TODO
  t_length[i] = # TODO

  # Transformed t interval
  t_med_cap[i] = # TODO
  t_med_length[i] = # TODO

  # 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)
  }

  t_boot_cap[i] = # TODO, use the df defined earlier
  t_boot_length[i] = # TODO

  perc_boot_cap[i] = # TODO
  perc_boot_length[i] = # TODO

  rev_boot_cap[i] = # TODO
  rev_boot_length[i] = # TODO
}
```

```
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)
```

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

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?