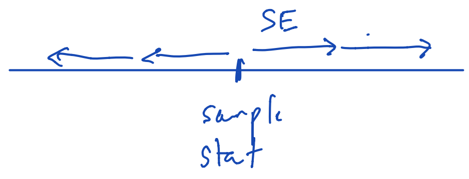
CI construction

2 approaches based on bootstrapping

1. Centered interval approach

$\boxed{\text{Use sample statistic as the center}}$ ($\bar{x}$, $\hat{p}$, $\bar{x}_1 - \bar{x}_2$, $\hat{p}_1 - \hat{p}_2$, r, b_1)



2. Use percentiles

95% CI : Go from 2.5-percentile to 97.5 percentile

80% CI : " " 10-percentile to 90 percentile

Is this a valid set of hypotheses

H_0 : parameter = null value

H_a : $p \neq 0.2$, $\mu \neq 98.6$, $\mu_1 - \mu_2 > 0$ } corresp. to these

{ $p = 0.2$, $\mu = 98.6$, $\mu_1 - \mu_2 = 0$

Not valid H_0: $\hat{p} = 0.2$ (statements should be about params., not statistics)

H_0: $\mu \leq 98.6$ (statement of equality in H_0, not inequality)

Stat 145, Wed 6-Oct-2021 -- Wed 6-Oct-2021
Biostatistics
Spring 2021

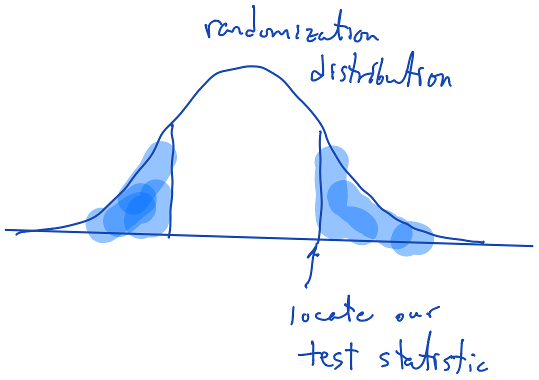
Wednesday, October 6th 2021

Wk 6, We
Topic:: Type I and II error
Topic:: Randomization distribution in RStudio
Read:: Lock5 4.3

Approximate null distribution — the distribution of values our sample statistic could take, along w/ frequency, when H_0 holds.

Review P-value
- meaning

The likelihood of seeing a test statistic at least as extreme as the one from our sample when H_0 is true.



- calculated as a proportion of values in
   two tail = twice what it is in one tail, use if alt hyp is "not equal"
   one tail, use if alt hyp is "greater than" or "less than"

- always between 0 and 1

Saying our P-value = 1.7 is nonsense

- the smaller it is
   the stronger the evidence against the null hypothesis
   the stronger the evidence in favor of alternative hypothesis

```
 - Acceptable/unacceptable language when result is NOT statistically significant
    Acceptable: We fail to reject the null hypothesis
    Unacceptable: We accept the null hypothesis

New concepts:
 - Type I and Type II error
    meaning
    Probability of committing Type I error is tied to alpha
```

Type I error: when H_0 is true, but P-value was used to reject it.

Setting $\alpha = 0.05$ corresponds to allowing a Type I error 5% of the time.

Type II error: when H_0 is false, but you don't conclude its false

Likelihood of Type II error rises as the likelihood (α) of Type I falls.

Randomization (i.e., approximate null) distributions in bivariate settings more interesting

Case: binary categorical expl. var., quantitative resp. var.

Treatment	Taps
caffeine	262
placebo	251
placebo	260
placebo	226
caffeine	342
⋮	⋮

used to calculate two sample means

$\bar{x}_p =$ ____

$\bar{x}_c =$ ____

like saying "no association"

$H_0: \mu_p - \mu_c = 0$

$H_a: \mu_p - \mu_c \neq 0$ (2-sided)

$H_a: \mu_p - \mu_c < 0$ reasonable

Call

$\bar{x}_p - \bar{x}_c$ our "test statistic"

Will locate this on null dist.

Case: Two quantitative vars.

If they appear to have a linear relationship

Might test

either

$H_0: \rho = 0$ or $H_0: \beta_1 = 0$

$H_a: \rho \neq 0$ $H_a: \beta_1 \neq 0$

$\rho > 0$ $\beta_1 < 0$

Some scenarios and types of hypotheses we might propose in them

1. **Univariate binary categorical data**

   Hypotheses: $\mathbf{H}_0: p = p_0$ (some number) vs. $\mathbf{H}_a: p \neq p_0$ (or 1-sided version)

   Common test statistics: X, the *count* of successes; $\widehat{p} = X/n$, the proportion of successes

   Useful R commands: Can avoid if you use StatKey

   For generating a single randomization statistic: `rflip()`, `rbinom()`

   ```
   rflip(50, prob=0.3)       # simulates 50 draws (called H or T) when p=0.3
   ```

   For generating a randomization distribution:

   ```
   manyRes <- do(5000) * rflip(50, prob=0.3)     # 5000 instances of a randomization stat
   ```

   For determining a P-value (after the command above):

   ```
   2*nrow( filter( manyRes, columnName >= testStatistic) ) / 5000   # tailor this to needs
   ```

2. **Quantitative response variable, two groups, independent samples**

   Hypotheses: $\mathbf{H}_0: \mu_1 - \mu_2 = 0$ vs. $\mathbf{H}_a: \mu_1 - \mu_2 \neq 0$ (or 1-sided version)

   Only reasonable test statistic: $\overline{x}_1 - \overline{x}_2$

   The layout of the data should be similar to that in **CaffeineTaps**, with one column/variable (explanatory) indicating the group for cases, and another indicating the (quantitative) response.

   ```
   head(sample(CaffeineTaps)[,1:2])

      Taps      Group
   12  245 NoCaffeine
   19  246 NoCaffeine
   5   248   Caffeine
   7   246   Caffeine
   6   250   Caffeine
   8   248   Caffeine
   ```

   Useful R commands: Can avoid if you use StatKey

   For generating the test statistic in original sample: `diffmean()`

   ```
   diffmean(responseVar ~ grpVariable, data=dataFrame)
   ```

   For generating a single randomization statistic: `shuffle()`

   ```
   diffmean(responseVar ~ shuffle(grpVariable), data=dataFrame)
   ```

   For generating a randomization distribution:

```
manyDiffs <- do(5000)*diffmean(responseVar~shuffle(grpVariable),data=dataFrame)
```

For determining a P-value (after the command above):

```
2*nrow( filter( manyDiffs, diffmean >= testStat ) ) / 5000   # tailor to needs
```

3. **Bivariate quantitative data, scatter plot displays linear pattern**

   Hypotheses (two options):

   $\mathbf{H}_0: \ \rho = 0$ vs. $\mathbf{H}_a: \ \rho \neq 0$ (or 1-sided version)

   $\mathbf{H}_0: \ \beta = 0$ vs. $\mathbf{H}_a: \ \beta \neq 0$ (or 1-sided version)

   test statistics: r (for correlation hypotheses), b (for slope hypotheses)

   Useful R commands (directed at correlation case): Can avoid if you use StatKey

   For generating the test statistic in original sample:

   ```
   cor(responseVar ~ explanatoryVar, data=dataFrame)
   ```

   For generating a single randomization statistic: `shuffle()`

   ```
   cor(responseVar ~ shuffle(explanatoryVar), data=dataFrame)
   ```

   For generating a randomization distribution:

   ```
   manyCors <- do(5000)*cor(responseVar ~ shuffle(explanatoryVar), data=dataFrame)
   ```

   For determining a P-value (after the command above):

   ```
   2*nrow( filter( manyCors, diffmean >= testStat ) ) / 5000   # tailor to needs
   ```

4. **Univariate quantitative data**

   Hypotheses: $\mathbf{H}_0: \ \mu = \mu_0$ (some number) vs. $\mathbf{H}_a: \ \mu \neq \mu_0$ (or 1-sided version)

   test statistics: $\bar{x}$

   Useful R commands [redacted] Can avoid if you use StatKey

   For generating the test statistic in original sample:

   ```
   original_xbar <- mean(variableInFocus, data=dataFrame)
   ```

   For generating a single randomization statistic: `resample()`

   ```
   mu0 <- valueFromNullHypothesis
   mean(variableInFocus, data=resample(dataFrame)) + mu0 - original_xbar
   ```

   For generating a randomization distribution:

   ```
   manyRecenteredMeans <- do(5000)*mean(variableInFocus, data=resample(dataFrame)) +
       mu0 - original_xbar
   ```

   For determining a P-value (after the command above):

```
2*nrow( filter( manyRecenteredMeans, mean >= original_xbar ) ) / 5000   # tailor to needs
```

5. **Quantitative response variable, two groups, paired samples**

   Hypotheses: $\mathbf{H}_0: \mu_{\text{Diff}} = 0$ vs. $\mathbf{H}_a: \mu_{\text{Diff}} \neq 0$ (or 1-sided version)

   Test statistic: $\bar{x}_{\text{Diff}}$

   The layout of the data should be similar to that in **Wetsuits**, where there are two response columns (quantitative) per case, one column representing one sample, and the other representing a second under (presumably) different conditions.

```
head(sample(Wetsuits))

   Wetsuit NoWetsuit Gender       Type orig.id
8     1.57      1.52      M triathlete       8
4     1.35      1.27      F triathlete       4
10    1.53      1.45      M triathlete      10
2     1.47      1.37      F triathlete       2
1     1.57      1.49      F    swimmer       1
12    1.51      1.41      M triathlete      12
```

   Idea: In the null distribution, any observed difference arises from randomness; it was just as likely that case's two values could have been labeled the opposite way, so that the sign of their difference could well have been opposite to what is observed.

   Note: StatKey does not have an app option to build a randomization distribution for matched pairs. You can find one, however, at this link

   Useful R commands: No counterpart in StatKey

   For generating the test statistic from original sample: `mutate()`

```
myData <- mutate(dataFrame, difference = qVar1 - qVar2)   # adds new column
mean(~difference, data=myData)
```

   For generating a single randomization statistic: `shuffle(), c()`

```
sampleSize = nrow(myData)
randomRoleSwap <- resample( c(-1,1), size=sampleSize )
mean(~(difference*randomRoleSwap), data=myData)
```

   For generating a randomization distribution:

```
manyDiffs <- do(5000) *
    mean(~(difference*resample(c(-1,1), size=sampleSize)), data=myData)
```

   For determining a P-value (after the command above):

```
2*nrow( filter( manyDiffs, mean >= testStat ) ) / 5000   # tailor to needs
```

6. **Bivariate binary categorical variables**, where one variable acts as a group identifier (explanatory), and the other is the response

Hypotheses: $\mathbf{H}_0: p_1 - p_2 = 0$ vs. $\mathbf{H}_a: p_1 - p_2 \neq 0$ (or 1-sided version)

Test statistic: $\widehat{p}_1 - \widehat{p}_2$

This is often the most complicated case to carry out in R, as you may only be given summary data, and have to make your own raw data set first. In the example below, I use the context of Examples 4.28–4.29 on p. 268. The data for that example comes from Table 4.9, but to make both variables binary we ignore all the "Desipramine" cases. You may well decide to use StatKey, and avoid RStudio, for these scenarios.

Useful R commands: Can avoid if you use StatKey

For preparing a data frame: `data.frame()`, `rbind()`

```
cokeAddiction <- rbind(
  do(18) * data.frame(group = "lithium", relapse = "yes", stringsAsFactors = TRUE),
  do(6) * data.frame(group = "lithium", relapse = "no", stringsAsFactors = TRUE),
  do(20) * data.frame(group = "placebo", relapse = "yes", stringsAsFactors = TRUE),
  do(4) * data.frame(group = "placebo", relapse = "no", stringsAsFactors = TRUE)
)
head(sample(cokeAddiction))
```

```
     group relapse .row .index orig.id
18 lithium     yes    1     18      18
10 lithium     yes    1     10      10
21 lithium      no    1      3      21
1  lithium     yes    1      1       1
12 lithium     yes    1     12      12
5  lithium     yes    1      5       5
```

For generating the test statistic from original sample: `diffprop()`

```
diffprop(relapse ~ group, data=cokeAddiction)
```

For generating a single randomization statistic: `shuffle()`

```
diffprop(relapse ~ shuffle(group), data=cokeAddiction)
```

For generating a randomization distribution:

```
manyDiffs <- do(5000) * diffprop(relapse ~ shuffle(group), data=cokeAddiction)
```

For determining a P-value (after the command above):

```
2*nrow( filter( manyDiffs, diffprop >= testStat ) ) / 5000   # tailor to needs
```