ae-23-starter

Two Categorical Variables: Case study: CPR and blood thinner

Cardiopulmonary resuscitation (CPR) is a procedure used on individuals suffering a heart attack when other emergency resources are unavailable. This procedure is helpful in providing some blood circulation to keep a person alive, but CPR chest compressions can also cause internal injuries. Internal bleeding and other injuries that can result from CPR complicate additional treatment efforts. For instance, blood thinners may be used to help release a clot that is causing the heart attack once a patient arrives in the hospital. However, blood thinners negatively affect internal injuries.

Here we consider an experiment with patients who underwent CPR for a heart attack and were subsequently admitted to a hospital. Each patient was randomly assigned to either receive a blood thinner (treatment group) or not receive a blood thinner (control group). The outcome variable of interest was whether the patient died within the 24 hours.

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v ggplot2 3.4.0     v purrr   0.3.5
v tibble  3.1.8     v dplyr   1.0.9
v tidyr   1.2.1     v stringr 1.4.1
v readr   2.1.3     v forcats 0.5.2

Warning: package 'ggplot2' was built under R version 4.2.2

Warning: package 'tidyr' was built under R version 4.2.2

Warning: package 'readr' was built under R version 4.2.2

Warning: package 'purrr' was built under R version 4.2.2
```

```
-- Conflicts --------------------------------------- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
-- Attaching packages ------------------------------------ tidymodels 1.0.0 --
v broom        1.0.1     v rsample      1.1.0
v dials        1.1.0     v tune         1.0.1
v infer        1.0.3     v workflows    1.1.0
v modeldata    1.0.1     v workflowsets 1.0.0
v parsnip      1.0.3     v yardstick    1.1.0
v recipes      1.0.3
```

```
Warning: package 'broom' was built under R version 4.2.2
```

```
Warning: package 'dials' was built under R version 4.2.2
```

```
Warning: package 'parsnip' was built under R version 4.2.2
```

```
Warning: package 'recipes' was built under R version 4.2.2
```

```
-- Conflicts --------------------------------------- tidymodels_conflicts() --
x scales::discard() masks purrr::discard()
x dplyr::filter()   masks stats::filter()
x recipes::fixed()  masks stringr::fixed()
x dplyr::lag()      masks stats::lag()
x yardstick::spec() masks readr::spec()
x recipes::step()   masks stats::step()
* Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(openintro)
```

```
Loading required package: airports
Loading required package: cherryblossom
Loading required package: usdata
```

```
Attaching package: 'openintro'

The following object is masked from 'package:modeldata':

    ames
```

```
data(cpr)
```

Remind ourselves of the situation:

Response Variable - outcome

Categorical Variable - group

Success - "died"

Ho: $\pi_{yes} - \pi_{no} = 0$

Ha:$\pi_{yes} - \pi_{no} \neq 0$

Calculate your sample statistic below. Next, write out your sample statistic using proper notation **(control - treatment)**. <- fix

```
stat_df <- cpr |>
  group_by(group, outcome) |>
  summarize(props = n()) |>
  pull(props)
```

```
`summarise()` has grouped output by 'group'. You can override using the
`.groups` argument.
```

```
control_stat <- stat_df[1]/(stat_df[1] + stat_df[2])
outcome_stat <- stat_df[3]/(stat_df[3] + stat_df[4])

obs_stat <- control_stat - outcome_stat

obs_stat
```

```
[1] 0.13
```

$\hat{p_{no}} - \hat{p_{yes}} = 0.13$

Conditions for the sampling distribution to be Normal

– Independence

– Success and failure condition

We use a special proportion called the pooled proportion to check the success-failure condition

p-pooled = Total number of successes (deaths) divided by the Total number of participants in the study

– Report this value below:

```
p_pool <- (stat_df[1] + stat_df[3]) / nrow(cpr) #nrow should be 90
```

This proportion is an estimate of survival rate across the study if the null hypothesis is true. We then use the following formula to check this condition:

$\hat{p_{pool}}$ x $n_1 > 10$?

$\hat{p_{pool}}$ x $n_2 > 10$?

$1 - \hat{p_{pool}}$ x $n_1 > 10$?

$1 - \hat{p_{pool}}$ x $n_2 > 10$?

Let's remind ourselves of each sample size...

```
cpr |>
  group_by(group) |>
  summarize(count = n())
```

```
# A tibble: 2 x 2
  group     count
  <fct>     <int>
1 control      50
2 treatment    40
```

The success-failure condition is satisfied since all values are at least 10.

Do we satisfy this condition?

Yes! We expect to see at least 10 successes and failures within each group

With both conditions satisfied, we can safely model the difference in proportions using a normal distribution.

Null standard error of the difference in two proportions, $\hat{p}_1 - \hat{p}_2$: =

Since we assume $\pi_1 = \pi_2$ when we conduct a theory-based hypothesis test for $H_0 : \pi_1 - \pi_2 = 0$, we substitute the **pooled sample proportion**, $\hat{p}_{pool}$ in for both π_1 and π_2 in the expression for the standard deviation of the statistic, resulting in its **null standard error**:

$$SE_0(\hat{p}_1 - \hat{p}_2) = \sqrt{\frac{\hat{p}_{pool}(1-\hat{p}_{pool})}{n_1} + \frac{\hat{p}_{pool}(1-\hat{p}_{pool})}{n_2}}$$
$$= \sqrt{\hat{p}_{pool}(1-\hat{p}_{pool})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}$$

This is the standard error formula we will use when computing the test statistic for a hypothesis test of $H_0 : \pi_1 - \pi_2 = 0$.

Calculate the standard error under the assumption of the null hypothesis below. Your calculation should come out to 0.0952

```
se <- sqrt(p_pool*(1-p_pool)*((1/50) + (1/40)))
```

The test statistic for assessing two proportions is a Z.

The Z score is a ratio of how the two sample proportions differ as compared to the expected variability of difference between the proportions.

$$Z = \frac{(\hat{p}_1 - \hat{p}_2) - 0}{\sqrt{\hat{p}_{pool}(1-\hat{p}_{pool})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

Using the above information, calculate our zscore:

```
zscore <- obs_stat / se
```

Using the normal distribution, calculate our p-value

```
2*pnorm(zscore, lower.tail = FALSE)
```

```
[1] 0.1712462
```

Why lower.tail = FALSE?

Because we want to look above our statistic and not below!

Why are we multiplying by 2?

Because we have a 2 sided test and are working with a symmetric distribution

Simpler Version: 1 Categorical Variable

Bumba or Kiki

How well can humans distinguish one "Martian" letter from another? In today's activity, we'll find out. When shown the two Martian letters, kiki and bumba, answer the poll https://app.sli.do/event/etoay5PwN5Mg5qiYg6BnDf whether you think bumba is option 1 or option 2.

Once it's revealed which option is correct, please write our sample statistic below:

.86

Let's write out the null and alternative hypotheses below

Ho: $\pi = 0.5$

Ha: $\pi > 0.5$

Now, let's quickly make a data frame of the data we just collected as a class. Replace the ... with the number of correct and incorrect guesses.

```
class_data <- tibble(
  correct_guess = c((rep("Correct" , 86)), rep("Incorrect" , 14))

)
```

Now let's simulate our null distribution by filling in the blanks. Below, detail how this distribution is created?

```r
set.seed(333)

null_dist <- class_data |>
  specify(response = correct_guess, success = "Correct") |>
  hypothesize(null = "point", p = 0.5) |>
  generate(reps = 1000, type = "draw") |>
  calculate(stat = "prop")
```

Helpful Hint: Remember that you can use `?` next to the function name to pull up the help file!

Set up a "spinner with half of it being correct and half being incorrect; spin it n = 100 times and record the new proportion of correct guesses

Do this above process 1000 times to create a distribution under the assumption of the null hypothesis!

Calculate and visualize the distribution below.

```r
visualize(null_dist) +
  shade_p_value(.86, direction = "right")
```

```
Warning in min(diff(unique_loc)): no non-missing arguments to min; returning Inf
```