hw3: A/B testing (permutation tests)

background reading:

- **A/B tests**
- **causality**

learning objectives:

- formulate competing hypotheses
- conduct A/B tests via simulation (resampling / permutation tests)
- explain and interpret p-values

name: Adrian

```
using DataFrames, CSV, PyPlot, Random
```

```
PyPlot.matplotlib.style.use("seaborn-bright")
```

modifying the Cavendish banana to make it more resistant to Panama disease

the Cavendish **banana is threatened** by Panama disease, which is caused by a fungus. if you are really interested in this problem, see **this YouTube video**. an active area of research is to genetically modify the Cavendish banana plant to make it resistant to Panama disease.

an experimental setup to determine if gene editing provides resistence to Panama disease

we have identified a gene in the banana plant, gene X, that is thought to be associated with resistance to fungal infection. using modern gene editing tools such as CRISPR, we are able to make a precise,

single-nucleotide edit to gene X that appears in the wild. the hope is that this single nucleotide polymorphism will confer protection against fungal infection.

we perform a randomized, controlled experiment to test whether this mutation of gene X confers greater resistance to Panama disease, or not.

- control group: 30 randomly selected banana plants (gene X wild type)
- treatment group: 30 randomly selected banana plants with gene X genetically modified.

none of these banana plants have been exposed to the fungus that causes Panama disease.

we then expose each plant in the experiment to the fungus that causes Panama disease. the plants are all contained in a single greenhouse. after one year, we record whether or not each grown banana plant is infected with Panama disease. the study is "blind" because the caretakers of the plants were never told which plants are genetically modified and which are not. this minimizes the chances that the caretakers (advertently or inadvertently) treat the two different variants of plants differently and thereby bias the outcome of the experiment.

🐸 the results of our randomized experiment are in `banana_study.csv`. each row represents a banana plant. the `:gene_x` attribute indicates whether gene X in the banana plant is the wild type or mutant. the `:outcome` attribute indicates the outcome of the experiment (infected or not infected). read in `banana_study.csv` as a `DataFrame`, `df`.

`df =`
60 rows × 2 columns

	gene_x String	outcome String
1	wild type	not infected
2	wild type	infected
3	wild type	not infected
4	wild type	not infected
5	wild type	not infected
6	wild type	not infected
7	wild type	infected
8	wild type	infected
9	wild type	not infected
10	wild type	not infected
11	wild type	not infected
12	wild type	not infected
13	wild type	not infected

	gene_x	outcome
	String	String
14	wild type	not infected
15	wild type	not infected
16	wild type	not infected
17	wild type	not infected
18	wild type	not infected
⋮	⋮	⋮

```
df = CSV.read("banana_study.csv", copycols=true)
```

🐸 use the `by` command to group the banana plants in `df` by genetic variant and create a new `DataFrame`, `df_outcome`, with a new column, named `p_infected`, that contains the proportion of plants of that variant that are infected by the fungus.

i.e. create a dataframe, `df_outcome`:

```
  gene_x      p_infected
wild type        ...
 mutant          ...
```

2 rows × 2 columns

	gene_x	p_infected
	String	Float64
1	wild type	0.133333
2	mutant	0.3

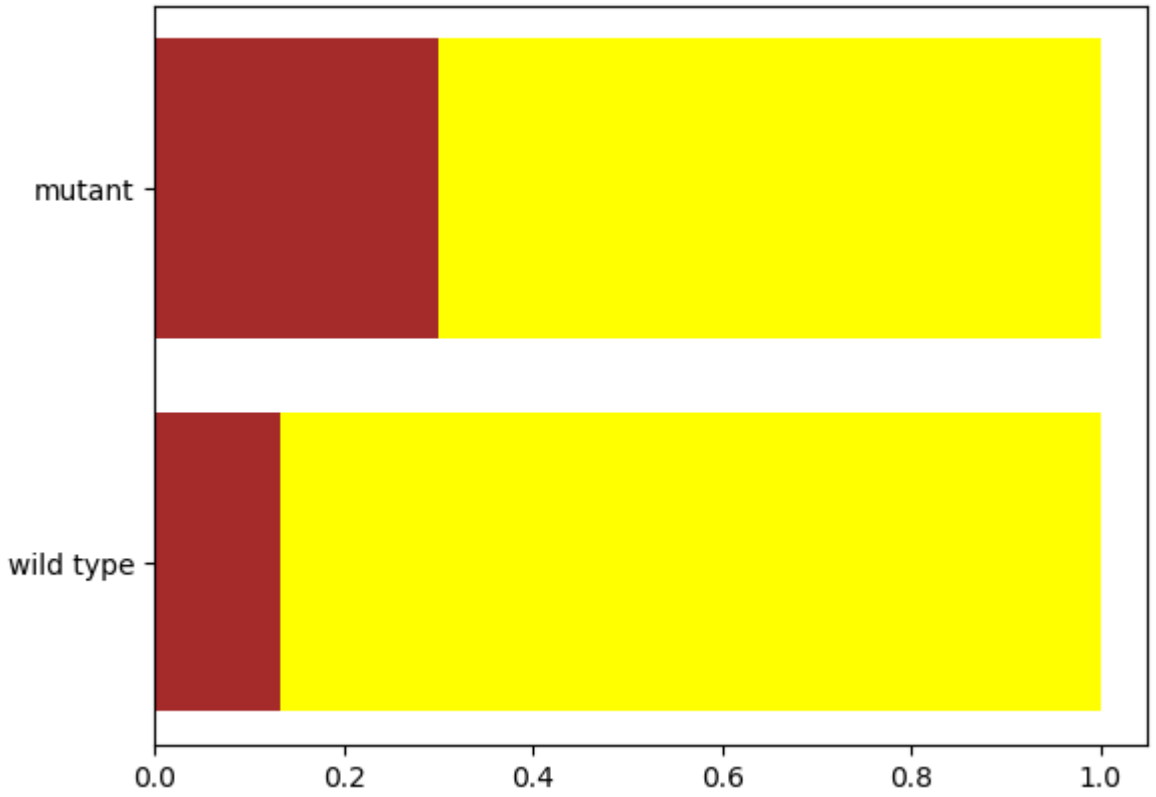
```
begin
    function infected_proportion(outcomes)
        infected = count(outcome -> (outcome == "infected"), outcomes)
        total = length(outcomes)
        return infected / total
    end

    df_outcome = by(df, :gene_x, p_infected=:outcome => infected_proportion)
end
```

🐸 create a data visualization of `df_outcome` above.

particularly, make a bar plot with two bars, one corresponding to wild type, the other to mutant. make the bar heights equal to the respective proportion of banana plants of that genetic variant that were infected by the fungus. with a different color, stack a bar *on top* of these bars to represent the

proportion that were not infected. the total height of each bar (sum of the two bars of different colors, stacked on top of each other) should be 1.0 to represent the entire set of bananas falling in that group. properly label your x-axis, y-axis, and x-ticks, and place a legend to denote the colors.

> **Hint**
>
> [illegible]




```
# could not get a plot using 'bottom' kwarg, but sequential plotting gets the job done

begin
    figure()
    barh(df_outcome.gene_x, 1, color="yellow")
    barh(df_outcome.gene_x, df_outcome.p_infected, color="brown")
    gcf()
end
```

🐸 which genetic variant (wild type vs. mutated) has the smallest proportion of infected plants?

why, then can't we conclude that this genetic variant is **certainly** more resistant to Panama disease than the other?

Wild type has a lower rate of infection *in this sampling*. It is possible that the observed difference is a statistical anomaly.

```
md"

```

```
Wild type has a lower rate of infection *in this sampling.*  It is possible that the
observed difference is a statistical anomaly.

"
```

🐸 formulate a null hypothesis and an alternative hypothesis that pertains to this experiment. state it below.

among the population of banana plants exposed to the fungus that causes Panama disease:

null hypothesis: mutation has no effect on infection rate

alternative hypothesis: mutation increases infection rate

```
md"
_null hypothesis_: mutation has no effect on infection rate

_alternative hypothesis_: mutation increases infection rate
"
```

🐸 define a test statistic that would differ depending on whether the null or alternative hypothesis were true. write your definition of the test statistic below.

> **Hint**
>
> a difference in proportions

test statistic:= $p(wildtype) - p(mutant)$ where $p(x)$ is the proportion of bananas in group x infected with Panama disease.

```
md"
_test statistic_:= $p(wild type) - p(mutant)$ where $p(x)$ is the proportion of
bananas in group $x$ infected with Panama disease.

"
```

🐸 write a function `proportion_infected` that takes in three arguments:

- `df_banana::DataFrame` : each row represents a 🦾 plant. has a column indicating which variant of gene X ("wild type" or "mutant") that plant harbors. also has an `:outcome` column ("infected" or "not infected"). your `df` will be passed in as this argument.
- `gene_x_col_name::Symbol` : the name of the column in `df_banana` that indicates which gene X variant the plants harbor ("wild type" or "mutant"). so `df_banana[1, gene_x_col_name]` gives us the gene X variant of banana 1.
- `gene_x::String` : either "wild type" or "mutant"

and returns the proportion of banana plants harboring `gene_x` (`="wild type"` or `="mutant"`) gene X that were infected by the fungus.

```
proportion_infected (generic function with 1 method)
```

```
function proportion_infected(df_banana::DataFrame, gene_x_col_name::Symbol, gene_x::String)
    outcomes = filter(row -> row[gene_x_col_name] == gene_x, df_banana).outcome
    infected = count(outcome -> (outcome == "infected"), outcomes)
    total = length(outcomes)
    return infected / total
end
```

🐸 test your function `proportion_infected` on the two different groups of plants.

```
0.13333333333333333
```

```
proportion_infected(df, :gene_x, "wild type")
```

```
0.3
```

```
proportion_infected(df, :gene_x, "mutant")
```

🐸 write a function `difference_in_proportions_infected` that takes in two arguments:

- `df_banana::DataFrame` as above
- `gene_x_col_name::Symbol` as above

and returns the test statistic.

> **Tip**
>
> inside this function, call `proportion_infected` twice

```
difference_in_proportions_infected (generic function with 1 method)
```

```
function difference_in_proportions_infected(df_banana::DataFrame, gene_x_col_name::Symbol)
    return proportion_infected(df_banana, gene_x_col_name, "wild type") - proportion_infected(df_banana, gene_x_col_name, "mutant")
end
```

🐸 using `difference_in_proportions_infected`, compute the actual, observed (as opposed to simulated under the null hypothesis, which we will do next) test statistic of our experimental outcome in `banana_study.csv` by passing in `df` as `df_banana` and `:gene_x` as `gene_x_col_name`. assign it to a variable to use later.

```
observed_δ = -0.16666666666666666
```

```
observed_δ = difference_in_proportions_infected(df, :gene_x)
```

🐸 imagine we are at the beginning of the experiment where we selected 60 banana plants. at this point, we are randomly selecting which banana plants receive the genetic modification on gene X and which do not. now, simulate one repetition of the banana study, *operating under the assumption that the*

null hypothesis is true, by permuting the labels in the `gene_x` column of `df`. this effectively simulates the random allocation of healthy banana plants to the two groups:

- group A: do not receive genetic modification on gene X (wild type)
- group B: receive genetic modification on gene X (mutant)

in this conceptual experiment, the outcome (infected vs. not) would be exactly the same *if the null hypothesis were true*, since then the genetic modification makes no difference in susceptibility to fungal infection. assign the permuted labels to be a new column in `df` called `:shuffled_gene_x`.

```
PooledArrays.PooledArray{String,UInt32,1,Array{UInt32,1}}: ["wild type", "mutant", "wi
```

```
df[!, :shuffled_gene_x] = shuffle(df.gene_x)
```

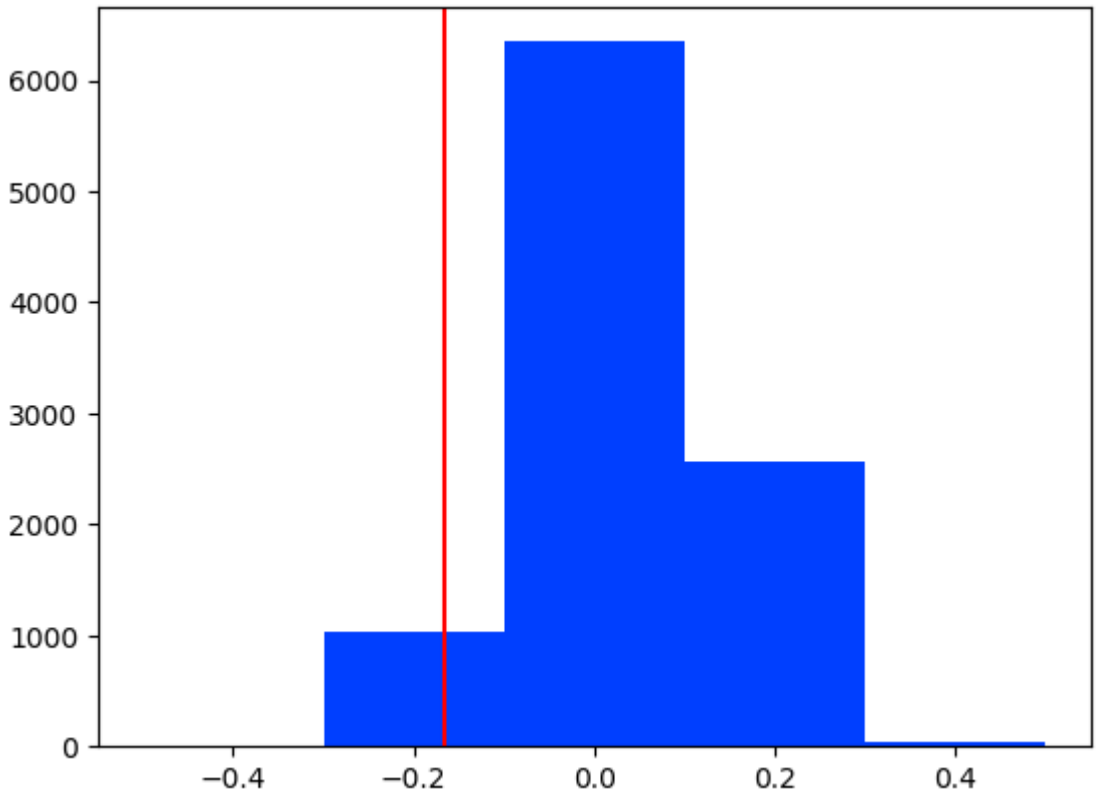
🐸 *operating under the assumption that the null hypothesis is true*:

(a) simulate 10,000 repetitions of the randomized banana experiment. keep track of the test statistic observed from each simulation by storing each test statistic in an array.

(b) plot the distribution of the test statistic. make your bins for the histogram manually to ensure the bin that includes zero is centered at zero. draw a red, vertical line at the actual test statistic observed in the experiment. the mean value of the test statistic should be around zero. make sure you understand why.

(c) compute the p-value associated with our null hypothesis that you obtained from your *random permutation test*.

```
begin
    simulated_stats = Float64[]
    for i in 1:1e4
        shuffle!(df.shuffled_gene_x)
        push!(simulated_stats, difference_in_proportions_infected(df,
:shuffled_gene_x))
    end
end
```




```
begin
    figure()
    hist(simulated_stats, bins=[-0.5:0.2:0.5...])
    axvline(observed_δ ,color="red")
    gcf()
end
```

```
p_value = count(stat -> stat ≤ observed_δ, simulated_stats)/length(simulated_stats);
```

🐸 explain what the the p-value represents.

the p-value is 0.1045

```
md"the p-value is $p_value"
```

🐸 if the level of significance is set at $\alpha = 0.05$, you may have the tendency to make a bold conclusion, such as "mutating gene X confers resistance to fungal infection" or "mutating gene X does not influence the susceptibility to fungal infection". however, such "dichotomization" is a mis-interpretation of the p-value. we must embrace uncertainty! see **this recent article in Nature** where, owing to the tendency to misinterpret p-values, some scientists call for abandoning the notion of statistical significance. When you've read the article, change the `Bool` below to `true`, and question (9) is complete! morever, statistical significance does not imply the effect is large!

> "We are calling for a stop to the use of P values in the conventional, dichotomous way – to decide whether a result refutes or supports a scientific hypothesis" **source**

you're done!

```
i_read_the_nature_article = true
```

- `i_read_the_nature_article = true`