homework 3: permutation tests - Vangj

```
using DataFrames
using CSV
using PyPlot
using Random

# (optional) change the style. see styles here: https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html
PyPlot.matplotlib.style.use("bmh")

# (optional) change settings for all plots at once, e.g. I like larger font size
rcParams = PyPlot.PyDict(PyPlot.matplotlib.rcParams)
rcParams["font.size"] = 16

# (optional) change the max width of columns of `DataFrame`
#  that are displayed in the Juptyer Notebook.
ENV["COLUMNS"] = 300 # characters
```

```
300
```

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. An active area of research is to genetically modify the Cavendish banana plant to make it resistant to Panama disease.

In this homework assignment, 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 the gene X that appears in the wild. The hope is that this single nucleotide polymorphism will confer protection against fungal infection.

To experimentally test whether a certain mutation of gene X will confer greater resistance to Panama disease, we randomly select 60 healthy banana plants from different regions/farms that have not been exposed to the fungus causing Panama disease (all free of Panama disease). Of these 60 plants, we randomly select 30 to receive the genetic mutation on gene X. We then collect one seed from all 60 plants.

A botanist then plants each seed in a pot whose soil is known to harbor the fungus that causes Panama disease. The plants are all contained in a single greenhouse. After one year, the botanist records whether or not each grown banana plant has been infected with Panama disease. The study is "blind" because we never told the botanist which seeds are modified and which are not. This way, the botanist cannot (advertently or inadvertently) treat the two different variants of plants differently and thereby influence the outcome of the experiment in a biased manner.

(1) 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 = CSV.read("banana_study.csv", copycols=true)
print(size(df)[1])
first(df,5)
```

```
60
```

5 rows × 2 columns

	gene_x	outcome
	String	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

(2) 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, `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         ...
```

```
df_outcomes = by(df, :gene_x, p_infected=:gene_x=>length)
```

2 rows × 2 columns

	gene_x	p_infected
	String	Int64
1	wild type	30
2	mutant	30

```
label = unique(df[!, :gene_x]) # testing purposes
label2 = unique(df[!, :outcome]) # testing purposes
```

```
2-element Array{String,1}:
 "not infected"
 "infected"
```

```
df_g = groupby(df, :gene_x) # testing purposes
# df_g[1, :gene_x]
# size(df_g[2])[1]
# size(df_g[!, :gene_x])
```

19	wild type	infected
20	wild type	not infected
21	wild type	not infected
22	wild type	not infected
23	wild type	not infected
24	wild type	not infected
25	wild type	not infected
26	wild type	not infected
27	wild type	not infected
28	wild type	not infected
29	wild type	not infected
30	wild type	not infected

⋮

Last Group (30 rows): gene_x = "mutant"

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

```
p_groups = groupby(df, :outcome)  # 2 groups by "not infected" and "infected"
# setting up list of infected and not infected
infected_list = []
not_infected_list =[]
for df_g in groupby(df, :gene_x)
    groub = df_g[1, :gene_x]
    infected = 0
    not_infected = 0
    for i in 1:length(df_g[!, :gene_x])
        testing = df_g[i, :outcome]
        if testing == "not infected"
            not_infected +=1
        else
            infected += 1
        end
    end
    push!(infected_list, infected)
    push!(not_infected_list, not_infected)
    println("For the group: ", groub)
    println("infected: ", infected)
    println("not infected: ", not_infected)
end
```

```
For the group: wild type
infected: 4
not infected: 26
For the group: mutant
infected: 9
not infected: 21
```

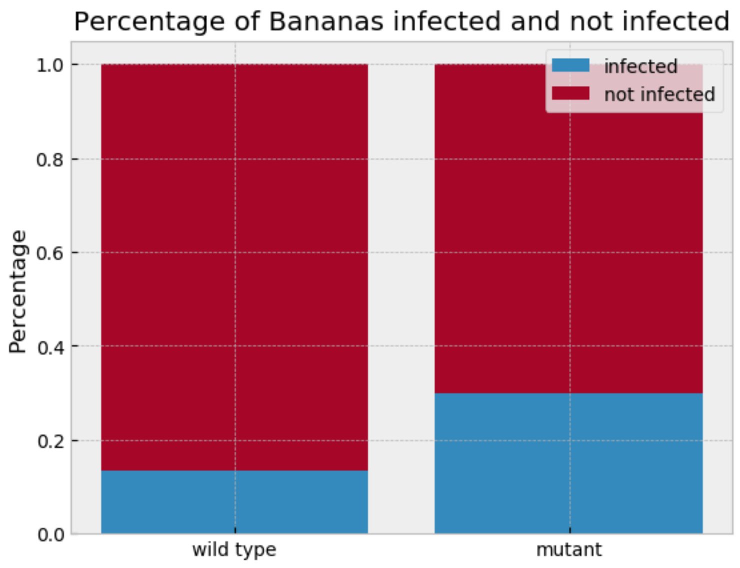
```
# Code determines the percentage of infected and not infected for both banana groups

p_infected = [NaN for s = 1:length(infected_list)]
p_not_infected = [NaN for s = 1:length(infected_list)]
for i in 1:length(infected_list)
    p_infected[i] = infected_list[i]/size(df_g[1])[1]
    p_not_infected[i] = not_infected_list[i]/size(df_g[1])[1]
    println("The Banana Group: ", label[i])
    println("the percentage infected is: ", p_infected[i])
    println("the percentage not-infected is: ", p_not_infected[i])
end
```

```
The Banana Group: wild type
the percentage infected is: 0.13333333333333333
the percentage not-infected is: 0.8666666666666667
The Banana Group: mutant
the percentage infected is: 0.3
the percentage not-infected is: 0.7
```

(3) create a data visualization representing `df_outcome`. particularly, make a bar plot with two bars, one corresponding to wild type, the other to mutant. Then 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. A skeleton code is shown below as an example. properly label your x-axis, y-axis, and x-ticks, and place a legend to denote the colors.

```
figure()
labels_gene = unique(df[!, :gene_x])
labels = unique(df[!, :outcome])
xticks(1:length(labels_gene), labels_gene)
ylabel("Percentage")
title("Percentage of Bananas infected and not infected")
bar(1:2, [p_infected[1], p_infected[2]], label = labels[2])
bar(1:2, [p_not_infected[1], p_not_infected[2]], bottom=[p_infected[1], p_infected[2]], label = labels[1])
# bar(1:2, [0.1333, 0.8666], label = labels[2])
# bar(1:2, [0.9, 0.5], bottom=[0.1, 0.5], label = labels[1])
legend()
```



```
PyObject <matplotlib.legend.Legend object at 0x00000000296B9668>
```

(4) Which genetic variant has the smallest proportion of infected plants? Can we conclude that this genetic variant is **certainly** more resistant to Panama disease than the other? If not, state a hypothetical but concrete reason why this outcome could be due to chance.

Response:

For this specific case, the wild type banana is more resistant to Panama disease, but we cannot conclude it is "certainly" more resistant with just 1 observation/test.

Some reasons that could of affected the results are that the soil could of have different concentrations of the Panama disease, outside forces like insects/animals that could move soil, attacks from pest, and the plants could of just been more resistive being grown in a greenhouse.

There are a lot of variants that could contribute to the spread of Panama disease and would need to conduct trails with a larger sample size and more known variables like Panama disease % in soil to conclude the results with certainty.

(5) Formally state the null hypothesis and alternative hypothesis that pertains to this experiment.

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

the **null hypothesis** is that the wild type of banana will be equally as resistive to the disease just like the mutant banana group. The difference in the percentages we see above is due to chance.

the **alternative hypothesis** is that the mutant banana type will act more postive or negative than the wild banana type. The distribution we see above is not due to chance. There is a direct correlation of cause and effect.

(6) Let us define and use the following test statistic, which would differ depending on whether the null or alternative hypothesis were true:

test statistic := (proportion of plants harboring mutant gene X that were infected) - (proportion of plants harboring wild type gene X that were infected)

(6a) write a function `proportion_infected` that takes in three arguments:

- `df_banana`, a `DataFrame` where each row is a banana, with an `:outcome` column (:infected" or "not infected") and another column indicating which variant of gene X ("wild type" or "mutant") that plant harbors. `df` from (1) is an example input argument.
- `gene_x_col_name`, a `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 variant of banana 1.
- `gene_x`, a `String`, that can be 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 fungus.

```
"""
return the proportion of banana plants infected by the fungus that harbor a certain gene X variant
"""
function proportion_infected(df_banana::DataFrame, gene_x_col_name::Symbol, gene_x::String)
    df_group = filter(df_banana -> df_banana[gene_x_col_name] == gene_x, df_banana)
    infected = 0
    for i in 1:size(df_group)[1]
        testing = df_group[i, :outcome]
        if testing == "infected"
            infected += 1
        end
    end
    return infected/size(df_group)[1]
end

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

```
    0.3
```

(6b) 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. Inside this function, you should call your function from (6a) directly above, twice.

compute the actual, observed (as opposed to simulated under the null hypothesis, which we will do next) test statistic of our data set using `df` from (1).

```
"""
p_wild_type := proportion of banana plants harboring wild type gene X infected by fungus
p_mutant := proportion of banana plants harboring mutant gene X infected by fungus

test statistic := p_mutant - p_wild_type.
"""
function difference_in_proportions_infected(df_banana::DataFrame, gene_x_col_name::Symbol)
    wild_banana_infected = proportion_infected(df_banana, gene_x_col_name, "wild type")
    mutant_banana_infected = proportion_infected(df_banana, gene_x_col_name, "mutant")
    return mutant_banana_infected - wild_banana_infected
end

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

```
    0.16666666666666666
```

(7) 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)

assume that the botanist plants these banana plant seeds in exactly the same pots in the greenhouse. 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`.

```
df[!, :shuffled_gene_x] = df[shuffle(1:nrow(df)), :gene_x]
first(df,6)
```

6 rows × 3 columns

	gene_x	outcome	shuffled_gene_x
	String	String	String
1	wild type	not infected	mutant
2	wild type	infected	wild type
3	wild type	not infected	mutant
4	wild type	not infected	mutant
5	wild type	not infected	mutant
6	wild type	not infected	wild type

```
difference_in_proportions_infected(df, :shuffled_gene_x)
```

```
0.033333333333333326
```

(8) **operating under the assumption that the null hypothesis is true:**

(8a) simulate 10,000 repetitions of the randomized banana experiment. keep track of the test statistic observed from each simulation.

(8b) 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.

(8c) compute the p-value associated with our null hypothesis that you obtained from your **random permutation test**.

(8d) precisely explain in a few sentences or less what this p-value represents.

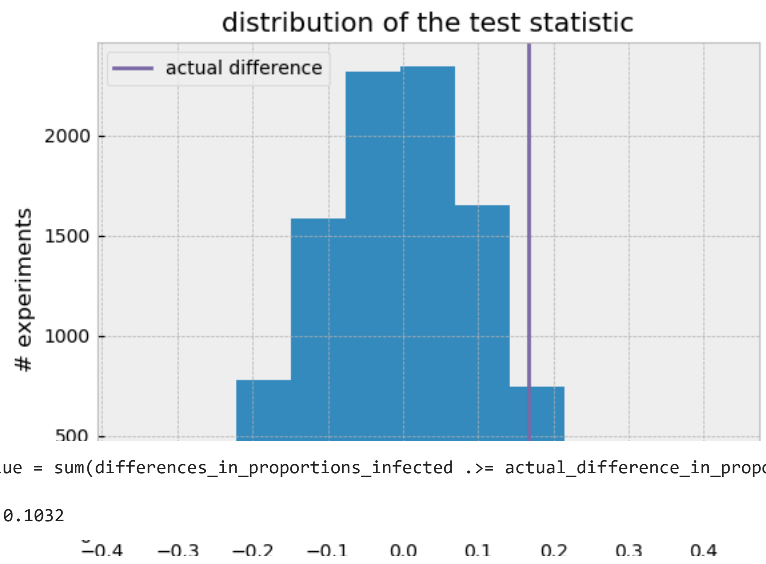
```
nb_sims = 10000 # number of simulations to run
# preallocate an array of NaNs in which we will store the resulting test statistics
differences_in_proportions_infected = [NaN for s = 1:nb_sims]
for s = 1:nb_sims
    # randomly permute groups = simulate random allocation of cauliflowers to groups
    df[!, :shuffled_gene_x] = df[shuffle(1:nrow(df)), :gene_x]
    # compute test statistic
    differences_in_proportions_infected[s] = difference_in_proportions_infected(df, :shuffled_gene_x)
end
first(df,6)
```

6 rows × 3 columns

	gene_x	outcome	shuffled_gene_x
	String	String	String
1	wild type	not infected	wild type
2	wild type	infected	wild type
3	wild type	not infected	mutant
4	wild type	not infected	wild type
5	wild type	not infected	wild type
6	wild type	not infected	mutant

```
# manually create bin so it is centered at zero
bins = range(minimum(differences_in_proportions_infected), stop=maximum(differences_in_proportions_infected), length = 12)

figure()
xlabel(L"$\mu_{wild_type} - \mu_{mutant}$")
ylabel("# experiments")
title("distribution of the test statistic")
# plot the actual test statistic as a vertical line
axvline(x=actual_difference_in_proportions_infected, color="C2", label ="actual difference" )
hist(differences_in_proportions_infected, bins=bins)
legend()
show()
```



```
p_value = sum(differences_in_proportions_infected .>= actual_difference_in_proportions_infected) / nb_sims
```

```
    0.1032
```

what is the p-value in this context?

Response:

the p-value indicates how many simulated experiments under the null hypothesis were equal or larger than the actual difference in proportions infected virus or what was observed in the actual experiment. In our case, it seems that 10% of the time we will reject the null hypothesis given it is true.

(9) 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.

> "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

warning

it is very easy to follow the steps and permute the labels in the `:gene_x` column without understanding *why*. please make sure to think carefully about why you can permute the labels to simualte the experiment under the null hypothesis. also think about why you cannot simulate the experiment under the alternative hypothesis.

random note on bananas

for those interested, here is an interesting, 20-minute YouTube documentary on the banana and how it is threatened by Panama disease.