Massachusetts Institute of Technology

Module Code:	15.077
Lecturer:	Roy Welsch
Coursework:	PS.03
Due Date:	25 March, 2021
Student Name:	Chun Hei (Samuel) LAM
MIT ID:	928931321

15.077 Statistical Learning and Data Science

Problem Sheet 3

Comparison of Two Samples

Declaration:

I pledge that the work submitted for this coursework is my own unassisted work unless stated otherwise.

Notes from Student:

Acknowledgement to Harry Yu. Please look at the python notebook attached if there are places that cannot be displayed properly.

[83]:

```
import numpy as np
from scipy.special import xlogy
from scipy import stats
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

[84]:

```
import plotly.express as px
```

1 Rice 10.48: Lottery for the Military Draft

In 1970, Congress instituted a lottery for the military draft to support the unpopular war in Vietnam. All 366 possible birth dates were placed in plastic capsules in a rotating drum and were selected one by one. Eligible males born on the first day drawn were first in line to be drafted followed by those born on the second day drawn, etc. The results were criticized by some who claimed that government incompetence at running a fair lottery resulted in a tendency of men born later in the year being more likely to be drafted. Indeed, later investigation revealed that the birthdates were placed in the drum by month and were not thoroughly mixed. The columns of the file `1970lottery` are month, month number, day of the year, and draft number.

[85]:

```
lottery = pd.read_csv("./1970lottery.txt", quotechar="'")
lottery1 = lottery[["Day_of_year", "Draft_No"]]
lottery2 = lottery[["Month_Number", "Draft_No"]]
```

[86]:

```
lottery1
```

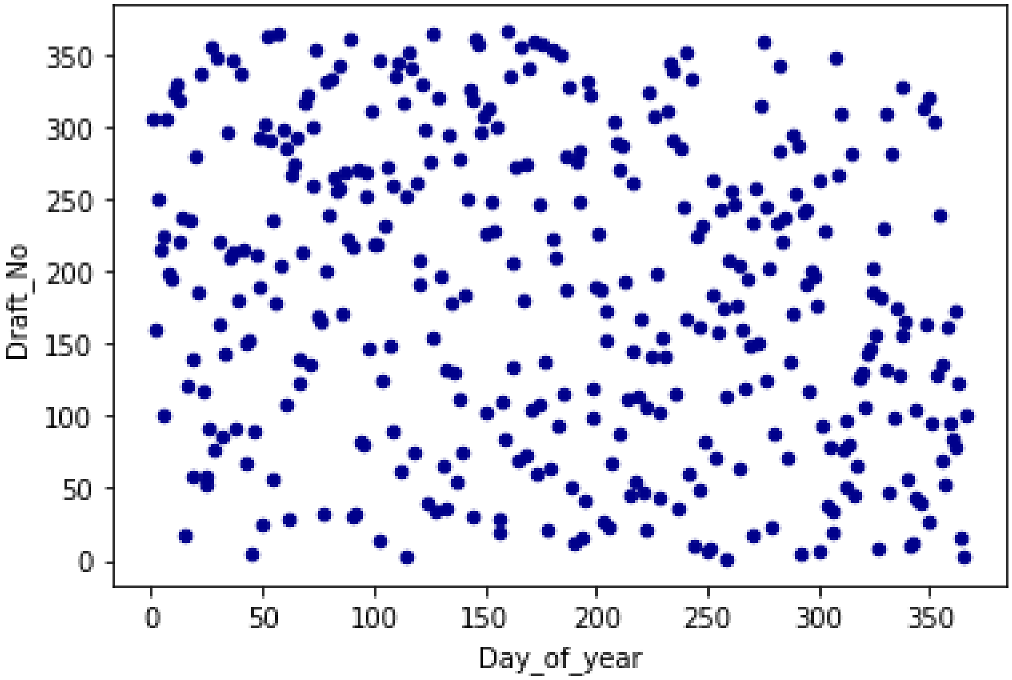
[86]:

```
     Day_of_year  Draft_No
0              1       305
1              2       159
2              3       251
3              4       215
4              5       101
..           ...       ...
361          362        78
362          363       123
363          364        16
364          365         3
365          366       100

[366 rows x 2 columns]
```

Part (a) Plot draft number versus day number. Do you see any trend?

[87]:

```
ax1 = lottery.plot.scatter(x='Day_of_year', y='Draft_No', c='DarkBlue')
```



Part (b) Calculate the Pearson and rank correlation coefficients. What do they suggest?

Solution: PMCC

```
[88]: pmcc = lottery1.corr()["Draft_No"]["Day_of_year"]
      pmcc
```

```
[88]: -0.2260414270110072
```

Spearman's Rank Correlation Coefficient

```
[89]: srcc = lottery1.corr(method="spearman")["Draft_No"]["Day_of_year"]
      srcc
```

```
[89]: -0.22580425074264954
```

Conclusion: They suggest there is a weak negative correlation between Draft_No and Day_of_year. It is less likely for men to be drafted at winter.

Part (c) *Is the correlation statistically significant?* One way to assess this is via a permutation test. Randomly permute the draft numbers and find the correlation of this random permutation with the day numbers. Do this 100 times and see how many of the resulting correlation coefficients exceed the one observed in the data. If you are not satisfied with 100 times, do it 1,000 times.

Solution:

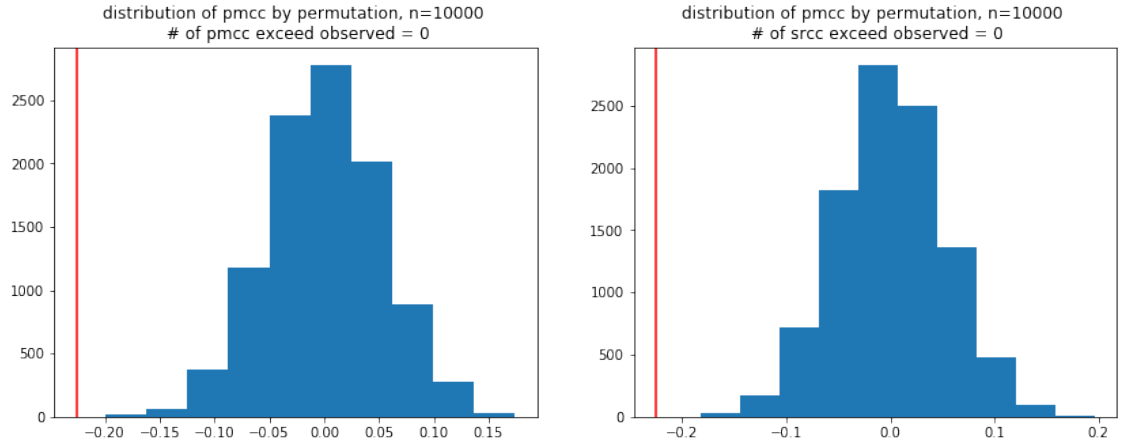
```
[90]: n = 10000
```

```
[91]: samp_corrcoef = [np.corrcoef(np.array(range(1,367)), np.random.
       ↪permutation(lottery1["Draft_No"]))[0][1] for i in range(n)]
```

```
[98]: samp_spearmanr = [stats.spearmanr(np.array(range(1,367)), np.random.
       ↪permutation(lottery1["Draft_No"]))[0] for i in range(n)]
```

```
[100]: fig, (ax1, ax2) = plt.subplots(1,2, figsize=(14,5))

       ax1.hist(samp_corrcoef)
       ax1.axvline(x=pmcc, color='r')
       ax1.set_title(f"distribution of pmcc by permutation, n={n} \n # of pmcc exceed␣
        ↪observed = {np.sum(np.absolute(samp_corrcoef)<srcc)}")

       ax2.hist(samp_spearmanr)
       ax2.axvline(x=srcc, color='r')
       ax2.set_title(f"distribution of pmcc by permutation, n={n} \n # of srcc exceed␣
        ↪observed = {np.sum(np.absolute(samp_spearmanr)<pmcc)}" )
```

```
[100]: Text(0.5, 1.0, 'distribution of pmcc by permutation, n=10000 \n # of srcc exceed
       observed = 0')
```



Comment: The histograms suggest that the observed $pmcc$ and $srcc$ are significant in the sense that the test $H_0 : \rho = 0$ against $H_0 : \rho \neq 0$ can be rejected with a very lower significant level (where ρ is population $pmcc$/$srcc$).

Part (d) Make parallel boxplots of the draft numbers by month. Do you see any pattern?

Solution: For this question, please open the attached $ipynb$ notebook to look at the interactive plot.

```
[101]: fig = px.box(lottery2, x="Month_Number", y="Draft_No")
       fig.show()
```

Pattern: we see that the median of `Draft_No` for different months are generally decreasing. The interquartile range of `Draft_No` are generally decreasing as well.

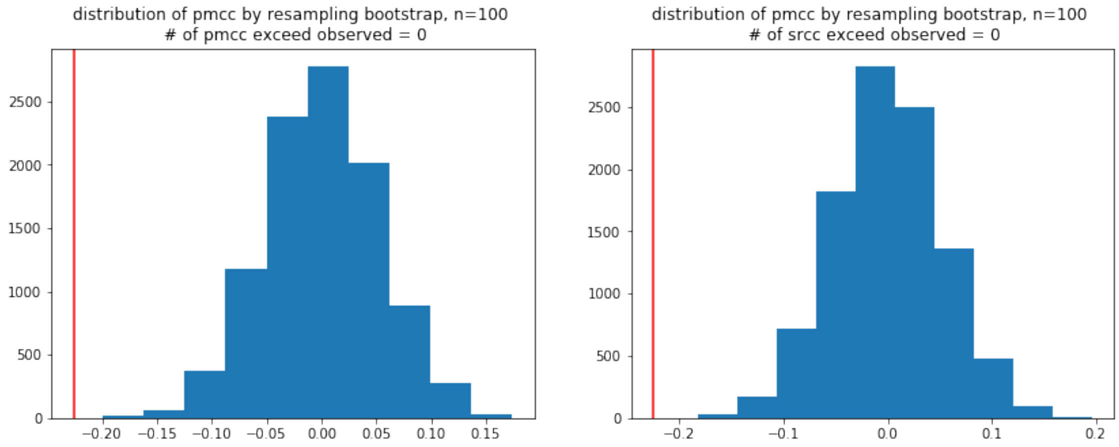
Part (e) Examine the sampling variability of the two correlation coefficients (Pearson and rank) using the bootstrap (re-sampling pairs with replacement) with 100 (or 1000) bootstrap samples. How does this compare with the permutation approach?

[102]:
```
n = 100
```

[103]:
```
boots_corrcoef = []
boots_spearmanr = []

for i in range(n):
    choice = np.random.choice(366,366)
    lottery_choice = [lottery1["Draft_No"].iloc[i] for i in choice]
    boots_corrcoef += np.corrcoef(choice+1, lottery_choice)[0][1]
    boots_spearmanr += stats.spearmanr(choice+1, lottery_choice)[0]
```

[105]:
```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(14,5))

ax1.hist(samp_corrcoef)
ax1.axvline(x=pmcc, color='r')
ax1.set_title(f"distribution of pmcc by resampling bootstrap, n={n} \n # of␣
 ↪pmcc exceed observed = {np.sum(np.absolute(samp_corrcoef)<srcc)}")

ax2.hist(samp_spearmanr)
ax2.axvline(x=srcc, color='r')
ax2.set_title(f"distribution of pmcc by resampling bootstrap, n={n} \n # of␣
 ↪srcc exceed observed = {np.sum(np.absolute(samp_spearmanr)<pmcc)}" )
```

[105]:
```
Text(0.5, 1.0, 'distribution of pmcc by resampling bootstrap, n=100 \n # of srcc
exceed observed = 0')
```



Comment: Again, the histograms suggest that the observed `pmcc` *and* `srcc` *are significant in the sense that the test $H_0 : \rho = 0$ against $H_0 : \rho \neq 0$ can be rejected with a very lower significant level (where ρ is population* `pmcc`*/*`srcc`*).*

2 Rice 11.15-16: Power of Comparison of Two Independent Samples:

Suppose that n measurements are to be taken under a treatment condition and another n measurements are to be taken independently under a control condition. It is thought that the standard deviation of a single observation is about 10 under both conditions.

Part (a) How large should n be so that a 95% confidence interval for $\mu_X - \mu_Y$ has a width of 2? (Use the normal distribution rather than the t-distribution, since n will turn out to be rather large.)

Solution: The length of the asymptotic confidence level is $2z_{0.05/2}\sigma_{\bar{X}-\bar{Y}}$, where $\sigma_{\bar{X}-\bar{Y}} \approx 10\sqrt{\frac{2}{n}}$. We want the length to be approximately equal to 2, therefore

$$2z_{0.05/2} \times 10\sqrt{\frac{2}{n}} = 2 \iff n = 200 \times z_{0.05/2}^2 \tag{1}$$

[106]:
```
z = stats.norm.ppf(1-(1-0.95)/2)
n = np.floor(200*(z**2) // 1)+1
print(f"n should be as large as {n}.")
```

```
n should be as large as 769.0.
```

Part (b) How large should n be so that the test of $H_0 : \mu_X = \mu_Y$ against the one-sided alternative $H_A : \mu_X > \mu_Y$ has a power of 0.5 if $\mu_X - \mu_Y = 2$ and $\alpha = 0.10$?

Solution: This indicates that $\mathbb{P}($Type II error$) = 0.5$, in otherwords, we have

$$\mathbb{P}\left(\frac{\bar{X}-\bar{Y}}{\sigma_{\bar{X}-\bar{Y}}} < z_{0.9} \,\middle|\, \mu_x - \mu_Y = 2\right) = 0.5$$
$$\implies \mathbb{P}\left(\frac{\bar{X}-\bar{Y}-2}{\sigma_{\bar{X}-\bar{Y}}} < z_{0.9} - \frac{2}{\sigma_{\bar{X}-\bar{Y}}} \,\middle|\, \mu_x - \mu_Y = 2\right) = 0.5$$

Of course, this further implies that

$$z_{0.9} - \frac{2}{\sigma_{\bar{X}-\bar{Y}}} = 0 \iff 10\sqrt{\frac{2}{n}} = \frac{2}{z_{0.9}} \iff n = 50z_{0.9}^2 \tag{2}$$

[107]:
```
z = stats.norm.ppf(1-0.1)
n = np.floor(50*(z**2))+1
print(f"n should be as large as {n}.")
```

```
n should be as large as 83.0.
```

3 Rice 11.39: Telephone Lines

An experiment was done to test a method for reducing faults on telephone lines (Welch 1987). Fourteen matched pairs of areas were used. The following table shows the fault rates for the control areas and for the test areas:

[108]:

```
telephone = pd.DataFrame({
    'Test' : [676,206,230,256,280,433,337,466,497,512,794,428,452,512],
    'Control' : [88,570,605,617,653,2913,924,286,1098,982,2346,321,615,519]})
telephone.T
```

[108]:

```
           0    1    2    3    4     5    6    7     8    9    10   11   12  \
Test     676  206  230  256  280   433  337  466   497  512   794  428  452
Control   88  570  605  617  653  2913  924  286  1098  982  2346  321  615

          13
Test     512
Control  519
```

Part (a) Plot the differences versus the control rate and summarize what you see.

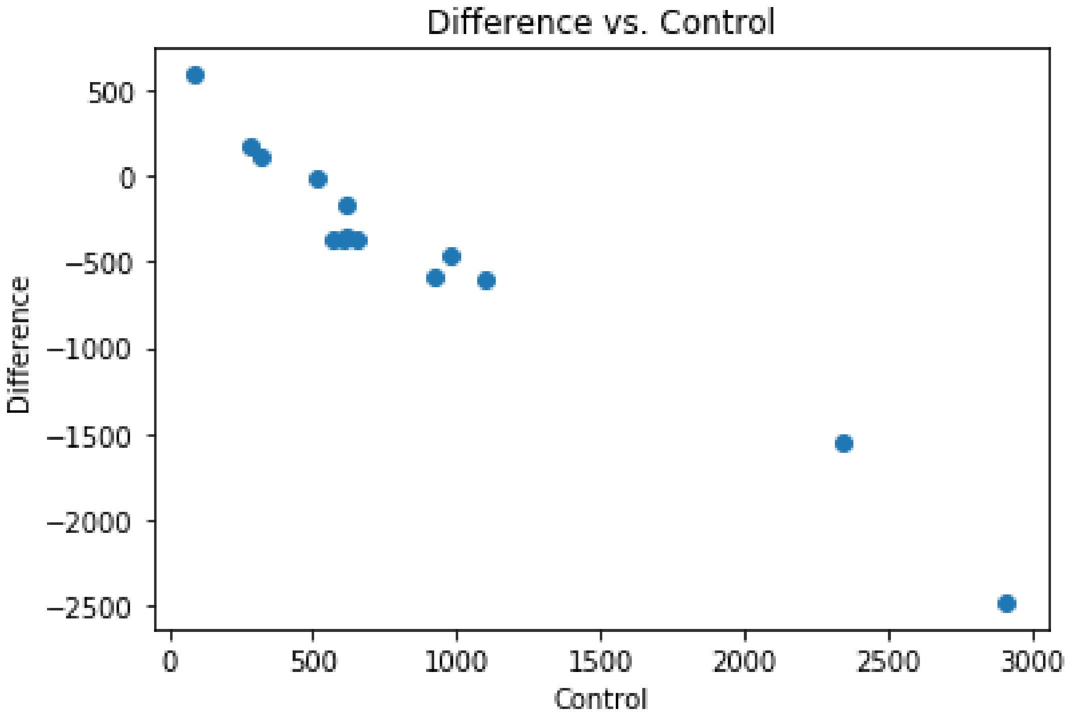
Solution: Plot

[109]:

```
plt.scatter(telephone.Control, telephone.Test - telephone.Control)
plt.xlabel("Control")
plt.ylabel("Difference")
plt.title("Difference vs. Control")
```

[109]:

```
Text(0.5, 1.0, 'Difference vs. Control')
```



Comment: seems like there is a positive correlation between the difference of rates and control rates - the higher the control rates are, the significant reduction of fault rates are.

Part (b) Calculate the mean difference, its standard deviation, and a confidence interval.

*Solution: We compute the difference.

[110]:

```
diff = telephone.Test - telephone.Control
```

First Calculate the mean and standard deviation.

[111]:

```
mean = diff.mean()
stdev = diff.std()
print(f"mean = {np.round(mean,3)}, standard deviation = {np.round(stdev,3)}")
```

```
mean = -461.286, standard deviation = 757.809
```

Then we compute the confidence interval.

[168]:

```
z1 = stats.t.ppf(0.025, df=14-1)
z2 = stats.t.ppf(0.975, df=14-1)
lower = np.round(mean+(stdev*z1/np.sqrt(14)),3)
upper = np.round(mean+(stdev*z2/np.sqrt(14)),3)
print(f"95% confidence interval is [{lower},{upper}]")
```

```
95% confidence interval is [-898.832,-23.74]
```

Part (c) Calculate the median difference and a confidence interval and compare to the previous result.

Solution: First compute the median.

[113]:
```
median = diff.median()
print(f"median = {median}")
```

```
median = -368.5
```

We choose $\alpha = 0.05$ *and construct a confidence interval. We first sort the data.*

[114]:
```
sorted_diff = diff.sort_values(ascending=True)
sorted_diff
```

[114]:
```
5    -2480
10   -1552
8     -601
6     -587
9     -470
2     -375
4     -373
1     -364
3     -361
12    -163
13      -7
11     107
7      180
0      588
dtype: int64
```

Then we find k *such that* $\mathbb{P}(N \leq k-1)$ *is closest to 0.025, where* $N \sim B(14, 0.5)$.

[115]:
```
stats.binom.ppf(0.025, 14, 0.5)
```

[115]:
```
3.0
```

[116]:
```
stats.binom.cdf(2, 14, 0.5), stats.binom.cdf(3, 14, 0.5), stats.binom.cdf(4,␣
 ↪14, 0.5)
```

[116]:
```
(0.006469726562499999, 0.02868652343750001, 0.08978271484375001)
```

Clearly $k = 3 + 1 = 4$ *is our optimal choice. Therefore the lower end-point is the 4-th entry, while the upper end-point is 14-4+1=11-th entry.*

[172]:
```
lower_b = sorted_diff.iloc[4-1]
upper_b = sorted_diff.iloc[11-1]
print(f"95% confidence interval is [{lower_b},{upper_b}]")
```

```
95% confidence interval is [-587,-7]
```

Comment: The non-parametric confidence interval is much wider since it assumes nothing about distribution of population. Turns out our data is more right-skewed, resulting in longer confidence interval.

Part (d) Do you think it is more appropriate to use a t-test or a nonparametric method to test whether the apparent difference between test and control could be due to chance? Why? Carry out both tests and compare.

Solution: Again set H_0 be there is no difference in rate, and H_1 be that there is (systematic) reduction in rate. Again we take $\alpha = 0.05$.

We first perform t-test. We compute the t statistic and its p-value.

[118]:

```
t = mean / (stdev/np.sqrt(14))
p = stats.t.cdf(t, df=14-1)

if p < 0.05:
    test_result = "reject H0."
else:
    test_result = "insufficient evidence to reject H0."

print(f"t statistic = {np.round(t,4)}, p-value = {np.round(p,4)}" + ", " +␣
 ↪test_result)
```

```
t statistic = -2.2776, p-value = 0.0201, reject H0.
```

Non-parametric method.

[119]:

```
w, p = stats.wilcoxon(diff, alternative="less", mode="exact")

if p < 0.05:
    test_result = "reject H0."
else:
    test_result = "insufficient evidence to reject H0."

print(f"Wilcoxon statistic = {np.round(w,4)}, p-value = {np.round(p,4)}" + ", "␣
 ↪+ test_result)
```

```
Wilcoxon statistic = 17.0, p-value = 0.0123, reject H0.
```

Comparision: both tests suggest there is sufficient evidence to reject `H0`. I would prefer exact Wilcoxon signed-rank test because it is more robust when handling data with outlier, like the one in our case.

4 Rice 11.46 - Cloud Seeding

The National Weather Bureau's ACN cloud-seeding project was carried out in the states of Oregon and Washington. Cloud seeding was accomplished by dispersing dry ice from an aircraft; only clouds that were deemed “ripe” for seeding were candidates for seeding. On each occasion, a decision was made at random whether to seed, the probability of seeding being $\frac{2}{3}$. This resulted in 22 seeded and 13 control cases. Three types of targets were considered, two of which are dealt with in this problem. Type I targets were large geographical areas downwind from the seeding; type II targets were sections of type I targets located so as to have, theoretically, the greatest sensitivity to cloud seeding. The following table gives the average target rainfalls (in inches) for the seeded and control cases, listed in chronological order. Is there evidence that seeding has an effect on either type of target?

```
[120]: control = pd.DataFrame({
           'Type I' : [0.0080,0.0046,0.0549,0.1313,0.0587,0.1723,0.3812,0.1720,0.
        ↪1182,0.1383,0.0106,0.2126,0.1435],
           'Type II' : [0.0000,0.0000,0.0053,0.0920,0.0220,0.1133,0.2880,0.0000,0.
        ↪1058,0.2050,0.0100,0.2450,0.1529]})
       control.head(4)
```

```
[120]:    Type I  Type II
       0  0.0080   0.0000
       1  0.0046   0.0000
       2  0.0549   0.0053
       3  0.1313   0.0920
```

```
[121]: seeded = pd.DataFrame({
           'Type I' : [0.1218,0.0403,0.1166,0.2375,0.1256,0.1400,0.2439,0.0072,0.
        ↪0707,0.1036,0.1632,0.0788,0.0365,0.2409,0.0408,0.2204,0.1847,0.3332,0.0676,0.
        ↪1097,0.0952,0.2095],
           'Type II' : [0.0200,0.0163,0.1560,0.2885,0.1483,0.1019,0.1867,0.0233,0.
        ↪1067,0.1011,0.2407,0.0666,0.0133,0.2897,0.0425,0.2191,0.0789,0.3570,0.0760,0.
        ↪0913,0.0400,0.1467]})
       seeded.head(4)
```

```
[121]:    Type I  Type II
       0  0.1218   0.0200
       1  0.0403   0.0163
       2  0.1166   0.1560
       3  0.2375   0.2885
```

Solution: we set H_0 be the hypothesis that there is no increase in rainfall, and H_1 be the hypothesis that there is an increase in rainfall.

For Type I target.

```
[176]: U, p = stats.mannwhitneyu(control["Type I"], seeded["Type I"],␣
        ↪alternative="less")

       if p < 0.05:
           test_result = "reject H0."
       else:
           test_result = "insufficient evidence to reject H0."
```

```
print(f"Mann-Whitney statistic = {np.round(U,4)}, p-value = {np.round(p,4)}" +␣
 ↪", " + test_result)
```

```
Mann-Whitney statistic = 130.0, p-value = 0.3348, insufficient evidence to
reject H0.
```

For Type II target.

[177]:

```
U, p = stats.mannwhitneyu(control["Type II"], seeded["Type II"],␣
 ↪alternative="less")

if p < 0.05:
    test_result = "reject H0."
else:
    test_result = "insufficient evidence to reject H0."

print(f"Mann-Whitney statistic = {np.round(U,4)}, p-value = {np.round(p,4)}" +␣
 ↪", " + test_result)
```

```
Mann-Whitney statistic = 108.0, p-value = 0.1194, insufficient evidence to
reject H0.
```

Conclusion: For both cases, there is insufficient evidence to conclude that there is an increase in rainfall due to cloud-seeding.

5 Rice 13.24 - Performance in Sporting Contest

Is it advantageous to wear the color red in a sporting contest? According to Hill and Barton (2005): > Although other colours are also present in animal displays, it is specifically the presence and intensity of red coloration that correlates with male dominance and testosterone levels. In humans, anger is associated with a reddening of the skin due to increased blood flow, whereas fear is associated with increased pallor in similarly threatening situations. Hence, increased redness during aggressive interactions may reflect relative dominance. Because artificial stimuli can exploit innate responses to natural stimuli, we tested whether wearing red might influence the outcome of physical contests in humans…

In the 2004 Olympic Games, contestants in four combat sports (boxing, tae kwon do, Greco-Roman wrestling, and freestyle wrestling) were randomly assigned red or blue outfits (or body protectors). If colour has no effect on the outcome of contests, the number of winners wearing red should be statistically indistinguishable from the number of winners wearing blue. They thus tabulated the colors worn by the winners in these contests:

[124]:

```
result = pd.DataFrame({
    "Sport": ["Boxing", "Freestyle Wrestling", "Greco Roman Wrestling", "Tae␣
 ↪Kwon Do"],
    "Red": [148, 27, 25, 45],
    "Blue": [120, 24, 23, 35]
})
```

```
result
```

[124]:

```
                     Sport  Red  Blue
0                   Boxing  148   120
1     Freestyle Wrestling   27    24
2   Greco Roman Wrestling   25    23
3              Tae Kwon Do   45    35
```

[125]:

```
row_total = result.Red + result.Blue
row_total
```

[125]:

```
0    268
1     51
2     48
3     80
dtype: int64
```

[126]:

```
col_total = result.sum().iloc[1:3]
col_total
```

[126]:

```
Red     245
Blue    202
dtype: object
```

[127]:

```
total = result.sum().iloc[1:3].sum()
total
```

[127]:

```
447
```

Some supplementary information is given in the file `red-blue.txt`.

Part (a) Let π_R denote the probability that the contestant wearing red wins. Test the null hypothesis $H_0 : \pi_R = 1/2$ versus the alternative hypothesis $H_1 : \pi_R$ is the same in each sport, but $\pi_R \neq 1/2$.

Solution: According to recitation, this is just a proportion test. Assume $\alpha = 0.05$. Under null hypothesis, the total number of winners wearing red (despite type of sport) follows the distribution $R \sim$ B$(447, 0.5)$, which can be approximated by a normal distribution N$(447 \times 0.5, 447 \times 0.25)$. We can now perform a two-sided z-test. Therefore the sample proportion approximately follows a N$(0.5, 0.25/447)$ distribution.

[169]:

```
z = ( (col_total.Red)/total - 0.5) / np.sqrt(1/ (total * 4))
p = 1 - stats.norm.cdf(z) + stats.norm.cdf(-z)

if p < 0.05:
    test_result = "reject H0."
else:
    test_result = "insufficient evidence to reject H0."
```

```
print(f"z statistic = {np.round(z,4)}, p-value = {np.round(p,4)}" + ", " +␣
 ↪test_result)
```

```
z statistic = 2.0338, p-value = 0.042, reject H0.
```

Part (b) Test the null hypothesis that $\pi_R = 1/2$ versus the alternative hypothesis that allows π_R to be different in different sports, but not equal to $1/2$.

Solution: We use a likelihood ratio test. According to recitation again, we can calculate the chi2 statistic as followed:

[129]:

```
X2_arr = 4*(result.Red - row_total/2)**2 / row_total
X2 = X2_arr.sum()
p = 1 - stats.chi2.cdf(X2,df=4)

if p < 0.05:
    test_result = "reject H0."
else:
    test_result = "insufficient evidence to reject H0."

print(f"chi2 statistic = {np.round(X2,4)}, p-value = {np.round(p,4)}, df = 4" +␣
 ↪", " + test_result)
```

```
chi2 statistic = 4.4352, p-value = 0.3503, df = 4, insufficient evidence to
reject H0.
```

Part (c) Are either of these hypothesis tests equivalent to that which would test the null hypothesis $\pi_R = 1/2$ versus the alternative hypothesis $\pi_R \neq 1/2$, using the data the total numbers of wins summed over all the sports?

Solution: This is equivalent to (a), but not (b).

Part (d) Is there any evidence that wearing red is more favorable in some of the sports than others?

Solution: We use χ^2 test for contingency table, which is the command $chi2\_contingency$ in the $scipy.stats$ library. Again we use $\alpha = 0.05$.

[130]:

```
chi2, pval, dof, expected = stats.chi2_contingency(result[["Red", "Blue"]].
 ↪values)

if p < 0.05:
    test_result = "reject H0."
else:
    test_result = "insufficient evidence to reject H0."

print(f"chi2 statistic = {np.round(chi2,4)}, p-value = {np.round(pval,4)}, df =␣
 ↪{dof}" + ", " + test_result)
```

```
chi2 statistic = 0.3015, p-value = 0.9597, df = 3, insufficient evidence to
reject H0.
```

As a reference, here the expected frequencies are tabulated.

```
[131]: expected
```

```
[131]: array([[146.89038031, 121.10961969],
              [ 27.95302013,  23.04697987],
              [ 26.30872483,  21.69127517],
              [ 43.84787472,  36.15212528]])
```

Part (e) From an analysis of the points scored by winners and losers, Hill and Barton concluded that color had the greatest effect in close contests. Data on the points of each match are contained in the file `red-blue.xls`. Analyze this data and see whether you agree with their conclusion.

Solution: First import the data.

```
[132]: boxing = pd.read_csv("red-blue_boxing.txt")
       freestyle = pd.read_csv("red-blue_FW.txt")
       grwrestling = pd.read_csv("red-blue_GR.txt")
       taikwondo = pd.read_csv("red-blue_TKD.txt")
```

We specifically look at the data when `Method of Win = "Points"` as these rounds represent "close contests". We therefore compute the difference of points (`Red` minus `Blue`) then filter out the `NaN`'s entries. For convenience we write a function that do our job.

```
[162]: def wilcoxon_sport(data):
           diff = (data["Points Scored by Red"] - data["Points Scored by Blue"]).
        ↪dropna()
           # diff_filtered = diff[diff <= diff.quantile(.25)]

           w, p = stats.wilcoxon(diff, alternative="greater")
           # w, p = stats.wilcoxon(diff_filtered, alternative="greater")

           if p < 0.05:
               test_result = "reject H0."
           else:
               test_result = "insufficient evidence to reject H0."

           print(f"Wilcoxon statistic = {np.round(w,4)}, p-value = {np.round(p,4)}" +␣
        ↪", " + test_result)
```

```
[163]: wilcoxon_sport(boxing)
```

```
Wilcoxon statistic = 13311.0, p-value = 0.3979, insufficient evidence to reject
H0.
```

```
[164]: wilcoxon_sport(freestyle)
```

```
Wilcoxon statistic = 835.5, p-value = 0.2105, insufficient evidence to reject
H0.
```

```
[165]: wilcoxon_sport(grwrestling)
```

```
Wilcoxon statistic = 580.5, p-value = 0.781, insufficient evidence to reject H0.
```

```
[166]: wilcoxon_sport(taikwondo)
```

```
Wilcoxon statistic = 1057.0, p-value = 0.4546, insufficient evidence to reject
H0.
```

For all sports, there is insufficient evidence to conclude that color has effect on the difference of points. Seems like the data does not agree with the conclusion.

Remark: 1. As pointed out by other classmates, this is not a good way to access whether color has *greatest* effect in close contest, because there are no other effects to be compared with. Perhaps we can access whether color has *significant* effect instead. In fact, according to `red-blue.txt`:

> For each sport, every contest was categorized on the basis of the quartile of the final points difference in the bout, where the first quartile of points difference represents symmetrical contests between competitors of similar ability and the fourth quartile represents contests between competitors with large asymmetries in ability. Contests stopped early (such as by knockout) were scored as highly asymmetric contests and coded in the fourth quartile.

There is no clear indication of quartiles. (The followed cell listed all labels in the datasets.) It is unclear whether we actually have to define the quantiles ourselves, so we adopt the simplest approach of dropping all `NaN` entries and do a Wilcoxon rank sum test (we did try to isolate the first quantile but has no progress). Appreciate your further guidance.

```
[145]: set(boxing["Method of Win"].values)
```

```
[145]: {'Disqualified',
         'Points',
         'Referee Stopped Contest',
         'Referee Stopped Contest - Outscored',
         'Walk Over'}
```

2. It is a little bit late to ask this, but it is not clear whether we need to manually code up all the testing algorithms instead of using built-in algorithms. Please clarify in lectures/recitations.