HW-3-dlhogan

October 24, 2021

1 CEWA 565 - HW-3

1.0.1 Daniel Hogan

1.0.2 10/26/2021

2 Problem 1

Download and work through the non-parametric tests notebook. Read the documentation and source code for the scipy.stats.ranksums and scipy.stats.mannwhitneyu functions.

Then answer the following questions:

A. What assumptions about our data and/or the hypothesis test are each of these two functions (ranksums and mannwhitneyu) making?

The ranksums function assumes that the samples provided are from continuous distributions. It does not handle ties between two sets of data. The ranksums calculation excludes the continuity correction that we applied in the non-parametric test code. It returns the Z statistic.

The mannwhitneyu function assumes the samples are independent from one another. It computes the U statistic and gives the option to include a continuity correction. U is computed in the same way we compute U in the non-parametric test code. In order to compute the Z statistic, W must be calculated from U and the Z statistic must be calculated from the computed W value.

B. Are there any additional inputs/options we need to specify for either or both functions to make sure that they duplicate our results from the non-parametric tests notebook?

We need to make sure that the alternative hypothesis value is setup correctly: by default both of these functions use two-sided tests. For both of these functions, the order of input determines how the statistics are calculated, so we also need to make sure the order is the same. The mannwhitneyu function offers the option to use a continuity correction of 1/2. If this value is set to false, the correction is not used and the outputted p-value is the same as the ranksums result. The method selection in the mannwhitneyu function depends on sample size. It is used to determine how the p-value is calculated. For large sample sizes (n>8 when'auto' method is used), p is calculated by comparing the standardized test statistic against the normal distribution and correcting for ties. The axis value is simply the axis along which the test is performed, 0 for rows, 1 for columns. We want axis to remain 0 since we are comparing the different data in each row (not data between columns). This parameter does not appear to be in the version of scipy that we have, however.

C. Revisit Homework 2 part D, using the observations of peak flow data for the Sauk

River to try and detect a change in streamflow around 1977. Perform the rank-sum test from Homework 2 part D again using the fuction(s) and/or options you identified here in part B. Discuss any differences in the test results that arise from slight differences in these two functions and the options you can choose.

[1]:
```
# import packages to be used
import numpy as np
import pandas as pd
import os
import scipy.stats as stats

import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
%matplotlib inline
```

[2]:
```
# change working directory
path = '/home/jovyan/Week-3/HW-3/'
os.chdir(path)
# save each river peak flow file
sauk_peaks = pd.read_excel('Sauk_peak_WY1929_2017.xlsx',skiprows=6).iloc[1:]
```

```
/opt/conda/lib/python3.8/site-packages/openpyxl/worksheet/_reader.py:312:
UserWarning: Unknown extension is not supported and will be removed
  warn(msg)
```

[3]:
```
# genarate pre and post 1977 Sauk River datasets
sauk_peaks_b = sauk_peaks[sauk_peaks['water_year']<1977]
sauk_peaks_a = sauk_peaks[sauk_peaks['water_year']>=1977]
# check that sample size is long enough
m = len(sauk_peaks_b)
n = len(sauk_peaks_a)
```

Compute the Z-stat and p-value using rank sums. This is the method I used in homework 2 part D.

[4]:
```
# Reference 2 - code adapted from lab 2-2
z_ranksum, p_ranksum = stats.ranksums(sauk_peaks_a['peak_va'],␣
 ↪sauk_peaks_b['peak_va'])
print("Z from stats.ranksums: {}".format(np.round(z_ranksum,4)))
print("P (two-sided) from stats.ranksums: {}".format(np.round(p_ranksum,4)))
```

```
Z from stats.ranksums: 2.4973
P (two-sided) from stats.ranksums: 0.0125
```

As expected, we get the same results as before.

Now, let's use the mannwhitneyu function to compute the same thing, setting this as a two-sided test (since we are just looking for any change in peak flows):

[111]:
```
# Compute Z and P using alternative = 'two-sided'
U_mannwhitneyu, p_mannwhitneyu = stats.mannwhitneyu(sauk_peaks_a['peak_va'],
                                                    sauk_peaks_b['peak_va'],
                                                    alternative='two-sided',
                                                    use_continuity=True)
# compute Z statistic from U value
W = U_mannwhitneyu + (n*(n+1))/2
Z = (W  - 0.5 -0.5*n*(m+n+1)) / np.sqrt( n*m*(n+m+1)/12 )
print("Z from stats.mannwhitneyu: {}".format(np.round(Z,4)))
print("P (two-sided) from stats.mannwhitneyu: {}".format(np.
 ↪round(p_mannwhitneyu,4)))
```

```
Z from stats.mannwhitneyu: 2.4934
P (two-sided) from stats.mannwhitneyu: 0.0126
```

Interesting… this change lead to a slighly lower z-stat and a slightly higher p-value, meaning this test result was, very slightly, closer to the null hypothesis of no change.

Now, let's remove the continuity correction from the U calculation and the calculation of Z and see what we get:

[112]:
```
# Compute Z and P using alternative = 'two-sided' and setting use_continuity to␣
 ↪False
U_mannwhitneyu, p_mannwhitneyu = stats.mannwhitneyu(sauk_peaks_a['peak_va'],
                                                    sauk_peaks_b['peak_va'],
                                                    alternative='two-sided',
                                                    use_continuity=False)
# compute Z statistic from U value
W = U_mannwhitneyu + (n*(n+1))/2
Z = (W -0.5*n*(m+n+1)) / np.sqrt( n*m*(n+m+1)/12 )
print("Z from stats.mannwhitneyu: {}".format(np.round(Z,4)))
print("P (two-sided) from stats.mannwhitneyu: {}".format(np.
 ↪round(p_mannwhitneyu,4)))
```

```
Z from stats.mannwhitneyu: 2.4973
P (two-sided) from stats.mannwhitneyu: 0.0125
```

This is the same as the ranksums result, meaning that this continuity correction for ties in rank accounts for the difference between these two results.

3 Problem 2

3.1 Part A.

First, plot the timeseries of the streamflow measurements as a function of water year for both watershed 1 and watershed 2 on the same graph. Use vertical dashed lines (axvline in matplotlib) to indicate the different periods (put a vertical dashed line in 1963, in 1967, and in 1982).

[80]:
```
# load in data
df = pd.read_excel('HJAndrews_peakflow_WS1_WS2_WS3.xlsx')
# data cleaning
df = df.rename(columns=df.iloc[1]).drop(df.index[0])
df = df.drop(df.columns[-1],axis=1)
df = df.rename(columns={df.columns[-1]:'period'})
df = df.drop(df.index[0]).reset_index()
df = df.drop(df.columns[0],axis=1).astype(float)
periods = ['control', 'active-logging', 'after-logging', 'long-after-logging']
for i,period in enumerate(periods):
    tmp = df['period'].copy(deep=True)
    tmp.loc[tmp==i+1] = period
    df['period'] = tmp
# check out result
df.head()
```

```
/opt/conda/lib/python3.8/site-packages/openpyxl/worksheet/_reader.py:312:
UserWarning: Unknown extension is not supported and will be removed
  warn(msg)
```

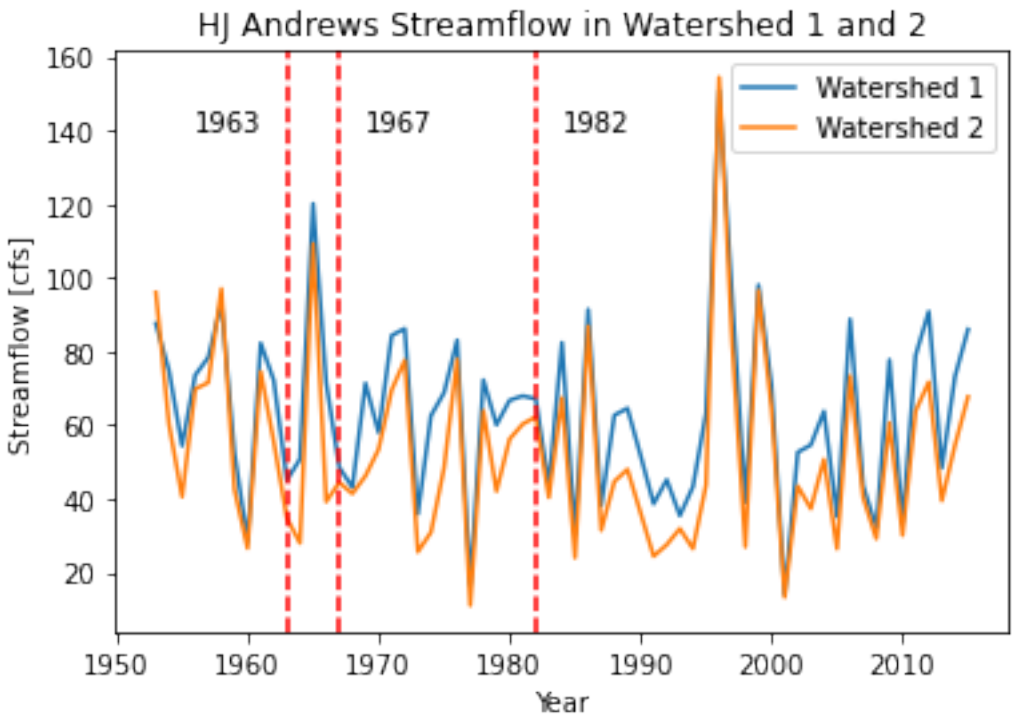
[80]:

	Water Year	Watershed 1	Watershed 2	Watershed 3	period
0	1953.0	87.5495	96.0073	81.9364	control
1	1954.0	74.7993	60.2205	50.7975	control
2	1955.0	54.4041	40.5364	35.1773	control
3	1956.0	73.5548	69.7704	54.6327	control
4	1957.0	78.3552	71.5483	57.1218	control

Now the data is cleaned up, let's plot up watershed 1 vs watershed 2 and add in the special dates

[81]:
```
# Watershed 1 plot
plt.plot(df['Water Year'],df['Watershed 1'], label = 'Watershed 1')
# Watershed 2 plot
plt.plot(df['Water Year'],df['Watershed 2'], label = 'Watershed 2')
# vertical lines for dates of interest
for year in [1963,1967, 1982]:
    plt.axvline(year, ls='--',color='r')
    if year == 1963:
        plt.text(year-7,140,str(year))
    else:
        plt.text(year+2,140,str(year))
plt.legend()
plt.xlabel('Year')
plt.ylabel('Streamflow [cfs]')
plt.title('HJ Andrews Streamflow in Watershed 1 and 2')
```

[81]:
```
Text(0.5, 1.0, 'HJ Andrews Streamflow in Watershed 1 and 2')
```



3.2 Part B.

It has been suggested that paired data such as this can be made to be closer to normally distributed by taking the log of each value before subtracting. Create two datasets: Q12 = streamflow1 - streamflow2 and Qlog12 = log(streamflow1)- log(streamflow2) and make graphs to demonstrate which is closer to normally distributed. Given that we want to use an ANOVA analysis, explain why is it important to do a transformation to get the data closer to normally distributed?

[82]:
```
# compute difference and log difference
q12 = df['Watershed 1'] - df['Watershed 2']
qlog12 = np.log10(df['Watershed 1'])-np.log10(df['Watershed 2'])
```

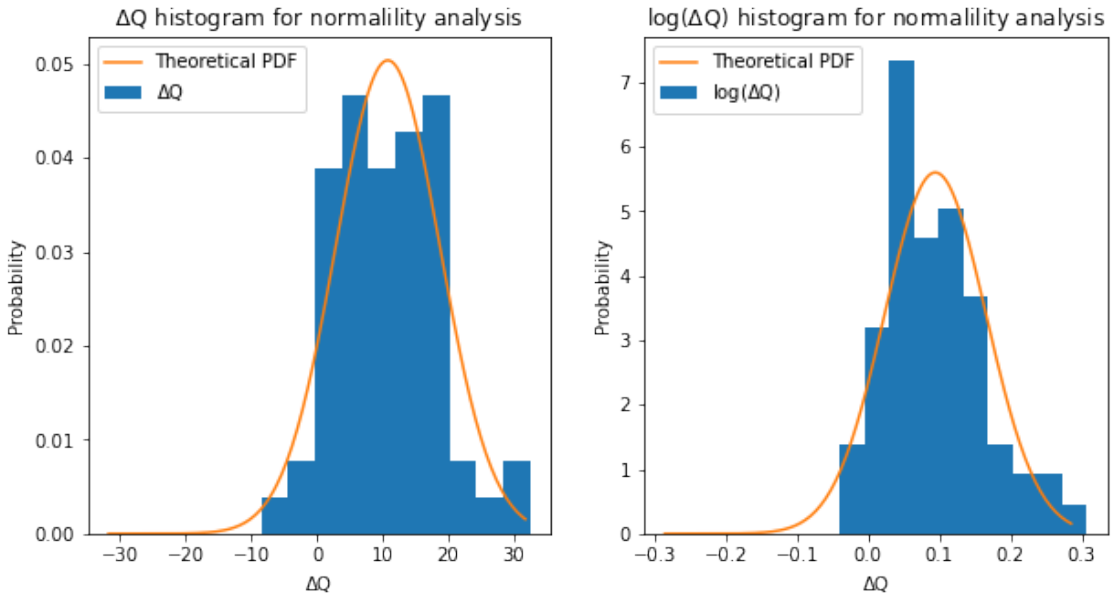
[83]:
```
nbins = 10
fig,axs = plt.subplots(ncols=2, figsize=(10,5))
axs[0].hist(q12,bins=nbins,density=True, label=r'$\Delta$Q') # to get pdfset␣
↪density=True
axs[1].hist(qlog12,bins=nbins,density=True, label=r'log($\Delta$Q)') # to get␣
↪pdfset density=True

# Create values for z
z_q = np.linspace(-4, 4, num=160) * q12.std()
z_qlog = np.linspace(-4, 4, num=160) * qlog12.std()
# Plot the theoretical PDF for each computation
```

```
axs[0].plot(z_q, stats.norm.pdf(z_q,q12.mean(),q12.std()), label='Theoretical␣
↪PDF')
axs[0].legend()
axs[0].set_ylabel('Probability')
axs[0].set_xlabel(r'$\Delta$Q')
axs[0].set_title('$\Delta$Q histogram for normality analysis')

axs[1].plot(z_qlog, stats.norm.pdf(z_qlog,qlog12.mean(),qlog12.std()),␣
↪label='Theoretical PDF')
axs[1].legend()
axs[1].set_ylabel('Probability')
axs[1].set_xlabel(r'$\Delta$Q')
axs[1].set_title('log($\Delta$Q) histogram for normality analysis')
```

[83]:
```
Text(0.5, 1.0, 'log($\\Delta$Q) histogram for normality analysis')
```



[84]:
```
# I choose the straight difference since it looks for normal to me. Now I add␣
↪this to the original dataframe
df['diff'] = q12
```

Although the log difference can improve normality, by comparing the plots above it looks like the simple difference method looks more normal when compared to a theoretical normal pdf. Thus, I will use the ΔQ difference method for my analysis.

The ANOVA test needs data that is normally distributed, so choosing a normalized dataset (or transforming it to look more normal) is vital for the ANOVA test to produce meaningful results.

3.3 Part C.

State the null and the alternative hypothesis for the question of whether the four periods are statistically different from each other. State the type I error (alpha value) that you are willing to accept.

Null Hypothesis: H_o: Difference in peak flows between WS1 and WS2 is the same for the four different time periods

Alternative Hypothesis: H_a: Difference in peak flows between WS1 and WS2 is statistically different for the four different time periods

I want 95% confidence, so $\alpha = 0.05$

In this case, we perform a one-way (also called one-factor) ANOVA to determine whether our null hypothesis (H_o) is true or not. We can reject the null hypothesis if $p < \alpha$.

3.4 Part D.

Perform an ANOVA test and discuss the results, related both to your hypothesis test listed above and to the more detailed question of which groups are statistically different from which other groups. Include graphs and/or tables that illustrate your results, and be sure to discuss what they mean. When using these ANOVA and other statistics functions, be sure that you understand what the code is doing (especially the defaults that different functions use) and outputting.

For this application, we need to make sure the following assumptions are valid for the p-value to be valid (taken from scipy.stats.f_oneway documentation: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html):

- The samples are independent. *This is true, the samples are not inter-dependent*
- Each sample is from a normally distributed population. *This is approximately true, we tried to find a transformation that would approximate normality*
- The population standard deviations of the groups are all equal. This property is known as homoscedasticity. *Since these are taken from the same population, the standard deviation is equal.*

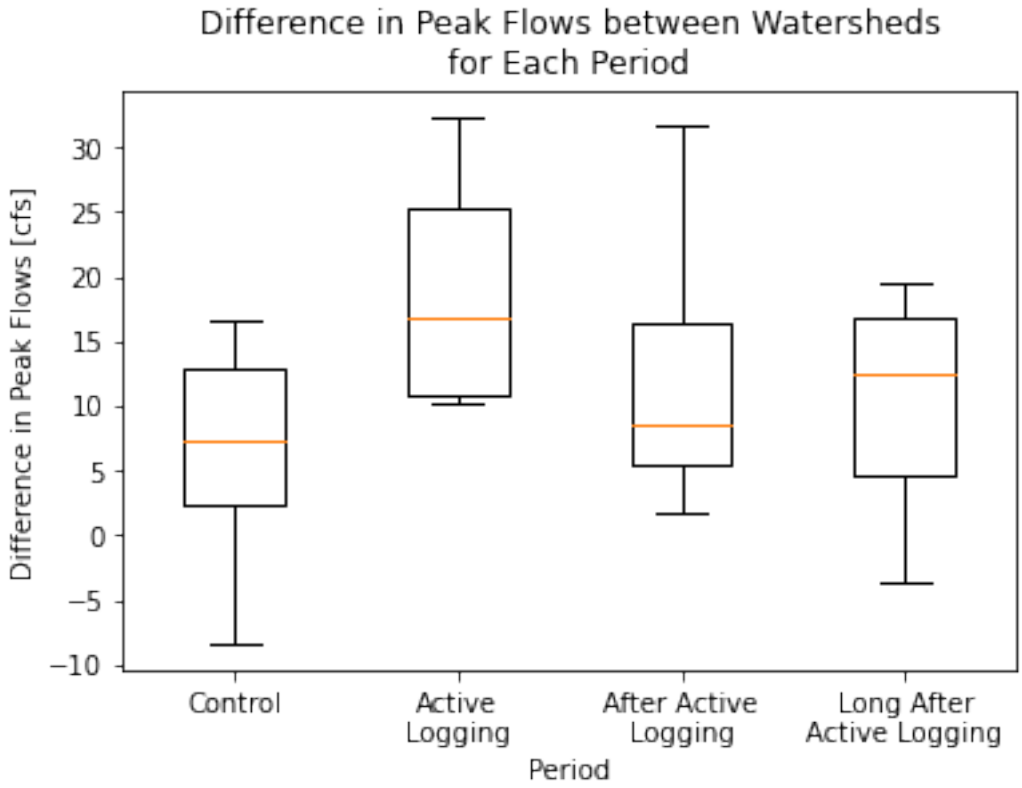
[108]:
```
# code adapted from lab 3-1
control = df[df['period']==periods[0]]
active = df[df['period']==periods[1]]
after_active = df[df['period']==periods[2]]
far_after_active = df[df['period']==periods[3]]
# Using boxplot, we can start to visually see differences between the periods
plt.boxplot([control['diff'],
             active['diff'],
             after_active['diff'],
             far_after_active['diff']],
            labels=['Control', 'Active\nLogging','After Active\nLogging','Long␣
 ↪After\nActive Logging'])

# Add labels
```

```
plt.xlabel('Period')
plt.ylabel('Difference in Peak Flows [cfs]')
plt.title('Difference in Peak Flows between Watersheds\nfor Each Period');
# stats f_oneway functions takes the groups as input and returns an F and␣
 ↪P-value
fvalue, pvalue = stats.f_oneway(control['diff'], active['diff'],␣
 ↪after_active['diff'], far_after_active['diff'])

# print the results
print("F-statistic = {}".format( np.round(fvalue,2)))
print("p = {}".format( pvalue ))
```

```
F-statistic = 2.87
p = 0.04402714850976671
```



This $p = 0.044$ is less than the chosen $\alpha = 0.05$, so we can reject the null hypothesis and accept the alternative, i.e. there was a change in peak flows between these time periods. The F statistic, where the p-value is derived from, is calculated by the following equation:

$$F = \frac{MSE}{MST}$$

where MSE is the variablity between groups and MST is the variability within a single group:

$$MSE = \frac{\sum_{j=1}^{k} n_j(\overline{y_j} - \overline{y})^2}{k-1}$$

$$MST = \frac{\sum_{j=1}^{k} \sum_{i=1}^{n_j}(y_{ij} - \overline{y_j})^2}{N-k}$$

Here, $1-k$ is the time period degrees of freedom (the number of sample means - 1) and $N-k$ is the error degrees of freedom (sum of the sample sizes, N, minus the number of samples,k). Now let's performa another ANOVA test using a different function:

[86]:
```
# following code blocks adapted from lab 3-1
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

[90]:
```
# performing this to show the other way the ANOVA test can be done with another␣
 ↪package
# reshape the d dataframe suitable for statsmodels package
df_reshaped = pd.concat([control[['period','diff']],
                         active[['period','diff']],
                         after_active[['period','diff']],
                         far_after_active[['period','diff']]])

# # Ordinary Least Squares (OLS) model - creates a linear fitted model between␣
 ↪diff and periods
# # this function takes an R style function where one column is fitted (~) to␣
 ↪another columns values
model = ols('diff ~ C(period)', data=df_reshaped).fit()
# print model summary results
print(model.summary())
# significance of typ=2: This type tests for the sum of squares of one event␣
 ↪after the other event.
#This is applicable in this case we are comparing events a
# from: https://md.psych.bio.uni-goettingen.de/mv/unit/lm_cat/
 ↪lm_cat_unbal_ss_explained.html
anova_table = sm.stats.anova_lm(model, typ=2)
# display the results table
anova_table
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   diff   R-squared:                       0.127
Model:                            OLS   Adj. R-squared:                  0.083
Method:                 Least Squares   F-statistic:                     2.868
Date:                Sat, 23 Oct 2021   Prob (F-statistic):             0.0440
Time:                        19:05:24   Log-Likelihood:                -214.97
No. Observations:                  63   AIC:                             437.9
```

```
Df Residuals:                       59   BIC:                             446.5
Df Model:                            3
Covariance Type:            nonrobust
==============================================================================
==================
                                   coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
------------------
Intercept                         19.1062      3.792      5.039      0.000
11.518      26.694
C(period)[T.after-logging]        -7.3246      4.268     -1.716      0.091
-15.864       1.215
C(period)[T.control]             -12.8226      4.487     -2.858      0.006
-21.801      -3.845
C(period)[T.long-after-logging]   -8.4253      4.009     -2.102      0.040
-16.447      -0.404
==============================================================================
Omnibus:                        1.361   Durbin-Watson:                   2.244
Prob(Omnibus):                  0.506   Jarque-Bera (JB):                1.234
Skew:                           0.189   Prob(JB):                        0.539
Kurtosis:                       2.427   Cond. No.                         9.64
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[90]:

	sum_sq	df	F	PR(>F)
C(period)	494.857606	3.0	2.867861	0.044027
Residual	3393.539593	59.0	NaN	NaN

Looks like values are the same, which they should be. The null hypothesis is still rejected.

Now let's see how each individual period is different from one another:

[88]:
```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
```

[95]:
```
# perform multiple pairwise comparison (Tukey HSD),
# endog is response variable, groups are the tested groups to compare
period_comp = pairwise_tukeyhsd(endog=df_reshaped['diff'],␣
 ↪groups=df_reshaped['period'], alpha=0.05)

# display the results table
print(period_comp)
```

```
           Multiple Comparison of Means - Tukey HSD, FWER=0.05
==============================================================================
```

```
    group1           group2       meandiff p-adj   lower    upper  reject
-----------------------------------------------------------------------------
active-logging      after-logging  -7.3246 0.3246  -18.608  3.9589  False
active-logging            control -12.8226 0.0292  -24.685 -0.9601   True
active-logging long-after-logging  -8.4253 0.1647 -19.0242  2.1737  False
 after-logging            control   -5.498 0.2953 -13.6839  2.6879  False
 after-logging long-after-logging  -1.1007    0.9  -7.3159  5.1145  False
       control long-after-logging   4.3973  0.381  -2.8159 11.6105  False
-----------------------------------------------------------------------------
```

The "reject" column in this table shows that the only periods that are different from one another are the active logging period and control period. All other periods were not shown to be statistically different from one another and, therefore, the null-hypothesis can be accepted for those periods (reject=False). In other words, by comparing the difference in watershed peak values in the active logging period and post logging periods and the post-logging periods and the control period, the Tukey test results in the table above show no statistical difference between these groups. The difference between the active-logging period and the control period, however, was significant and the null hypothesis is rejected.

3.5 References:

- Lab 2-2 available here:https://mountain-hydrology-research-group.github.io/data-analysis/modules/lab2/lab2-2.ipynb
- Lab 3-1 available here: https://mountain-hydrology-research-group.github.io/data-analysis/modules/lab3/lab3-1.ipynb
- Information on assumptions needed for valid p-value from ANOVA test: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html
- Information on typ value for anova table from: https://md.psych.bio.uni-goettingen.de/mv/unit/lm_cat/lm_cat_unbal_ss_explained.html

[113]:
```
!jupyter nbconvert --to pdf HW-3-dlhogan.ipynb
```

```
[NbConvertApp] Converting notebook HW-3-dlhogan.ipynb to pdf
[NbConvertApp] Support files will be in HW-3-dlhogan_files/
[NbConvertApp] Making directory ./HW-3-dlhogan_files
[NbConvertApp] Making directory ./HW-3-dlhogan_files
[NbConvertApp] Making directory ./HW-3-dlhogan_files
[NbConvertApp] Writing 62534 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 159066 bytes to HW-3-dlhogan.pdf
```

[ ]: