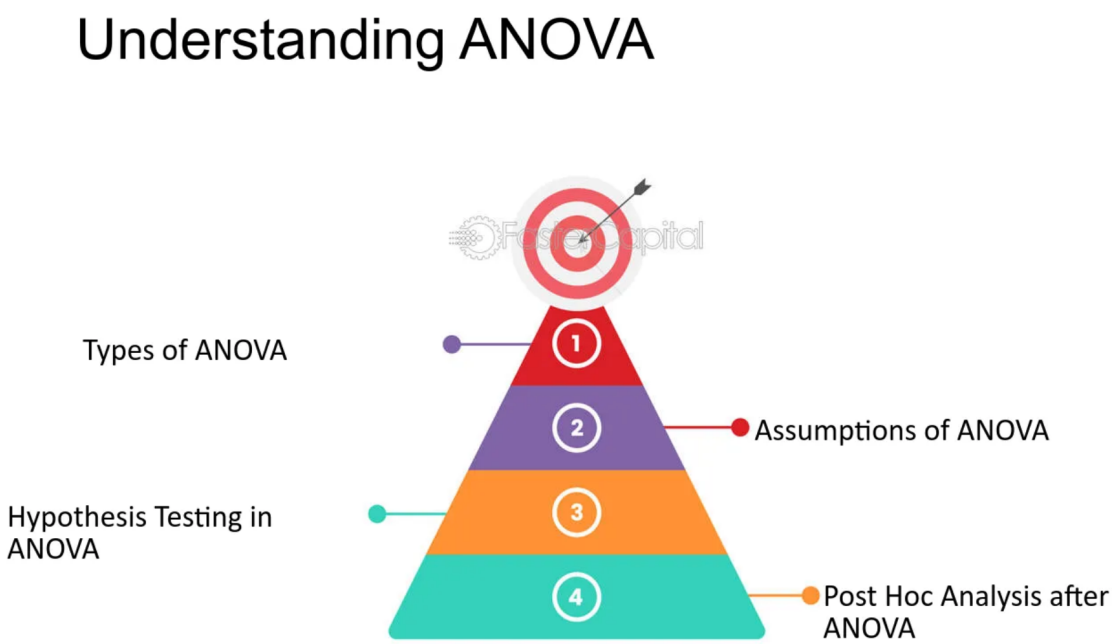
Comparing Centers of Several Independent Groups

EN5423 | Spring 2024

w14_several_independent_03.pdf
(Week 14)

Contents



1 Two-factor Permutation Test

• For two-factor and more complex ANOVAs where the data within one or more treatment groups are not normally distributed and may not have equal variances, permutation (also called randomization) tests can evaluate both factor effects and interactions.

• Manly (2007) evaluated several proposed methods for randomizing observations or randomizing residuals by subtracting the a · b group means, producing *p*-values more robust to violations of assumptions than classic ANOVA *F*-tests.

• Unlike nonparametric tests or ANOVA tests on logarithms, *permutation tests can determine whether group means differ as a result of factor effects* rather than group percentiles or geometric means. If the factor effects on group means are of interest, the permutation test should be preferred over classical two-factor ANOVA when non-normality or unequal variance appear.

Example 1. Iron at low flows—Two-actor permutation test.

A two-factor permutation test on iron concentrations is performed using the provided Python script. The current default of 100 permutation rearrangements is much too small. Using 5,000 rearrangements only takes a few seconds, yet it provides much greater precision and therefore repeatability of the resulting *p*-values.

```
data = pd.DataFrame({
    'fe': data_iron.Fe,  # iron concentration
    'mining': data_iron.Mining,
    'rocktype': data_iron.Rocktype
})

# Define the function for calculating F-statistic
def calculate_f_statistic(data, factor1, factor2, response):
    model = ols(f'{response} ~ {factor1}*{factor2}', data=data).fit()
    anova_results = sm.stats.anova_lm(model, typ=2)
    return anova_results['F']

# Function to perform the permutation test
def two_factor_permutation_test(data, factor1, factor2, response,
num_permutations=100):
    observed_f_stat = calculate_f_statistic(data, factor1, factor2,
response)

    def permuted_stat():
        permuted_data = data.copy()
        permuted_data[response] =
np.random.permutation(permuted_data[response].values)
        return calculate_f_statistic(permuted_data, factor1, factor2,
response)

    perm_stats = np.array([permuted_stat() for _ in
range(num_permutations)])
```

```
    p_values = {
        factor1: np.mean(perm_stats[:, 0] >= observed_f_stat.iloc[0]),
        factor2: np.mean(perm_stats[:, 1] >= observed_f_stat.iloc[1]),
        f'{factor1}:{factor2}': np.mean(perm_stats[:, 2] >=
observed_f_stat.iloc[2])
    }

    return observed_f_stat, p_values

# Perform the test
observed_f_stat, p_values = two_factor_permutation_test(data, 'mining',
'rocktype', 'fe', num_permutations=5000)

print("Observed F-statistics:", observed_f_stat)
print("P-values:", p_values)
>>
Observed F-statistics:
mining             2.492636
rocktype           2.379906
mining:rocktype    1.997428
Residual                NaN
Name: F, dtype: float64
P-values: {'mining': 0.0006, 'rocktype': 0.0102, 'mining:rocktype':
0.0266}
```

Note: Permuting the data in the context of a permutation test ***breaks any existing relationship*** between the response variable (fe) and the factors (mining and rocktype). This allows us to assess how unusual the observed data is under ***the null hypothesis that there is no effect of the factors on the response variable***.

• X1 is the generic name assigned to the first factor listed in the input command (mining).

• X2 is the generic name assigned to the second factor listed (rocktype).

• The interaction term is listed as X1:X2.

• Although ANOVA was unable to find significance for either factor or for the interaction, ***all three tests are significant using the permutation test***.

• This agrees with what was obvious to the eye in **Figures 1** and **2** below, where group boxplots clearly differ for the three mining history categories, and the interaction plot of **Figure 3** exhibited a large difference in mean concentration for sandstone as compared to limestone for the abandoned mine sites.

• The loss in power resulting from the assumptions required for classic ANOVA is real and common for environmental data, as our field data is usually much more strongly skewed and non-normal than those of more controlled experiments in other disciplines.

• Permutation tests are vital, therefore, to not miss effects that are actually present in data.

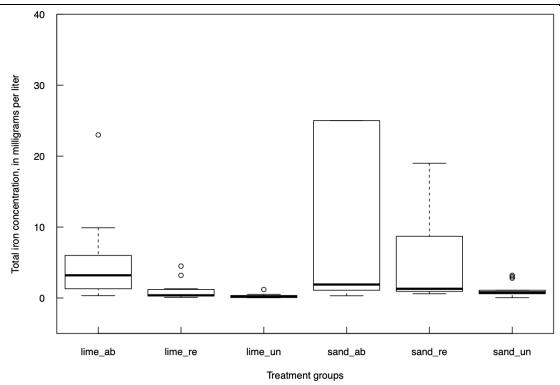


Figure 1. Boxplots of iron concentrations at low flow from Helsel (1983). Three outliers greater than 100 milligrams per liter are not shown. Lime_ab, limestone, abandoned mine; lime_re, limestone, reclaimed; lime_un, limestone, unmined; sand_ab, sandstone, abandoned mine; sand_re, sandstone, reclaimed; sand_un, sandstone, unmined.

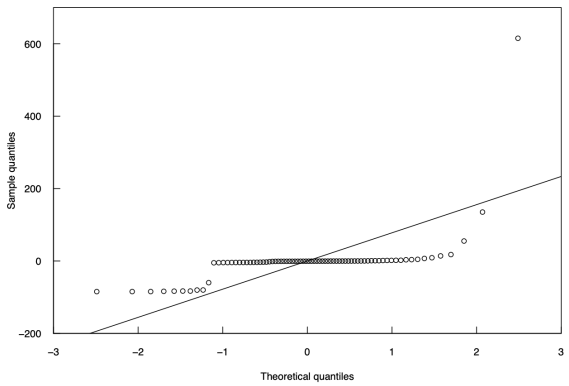


Figure 2. Q-Q plot showing the non-normality of the ANOVA residuals of the iron data.

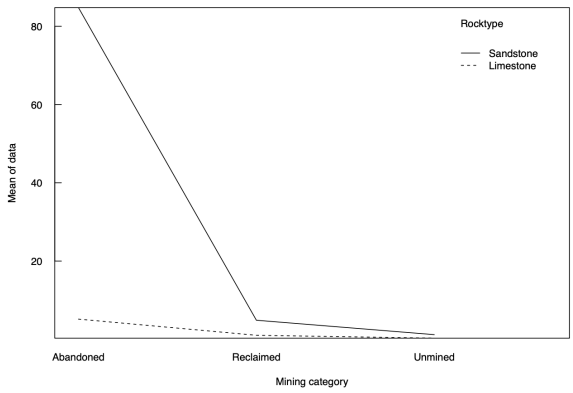


Figure 3. Interaction plot showing interaction by the large nonparallel increase in the mean for the combination of abandoned mining history and sandstone rock type.

2 Two-factor Nonparametric Brunner-Dette-Munk (BDM) Test

• The BDM test is a nonparametric test for multi-factor ANOVA designs that is ***particularly adept at dealing with heteroscedastic data (i.e., unequal variance)***.

• The test determines whether the frequencies of high versus low values have dissimilar patterns attributable to two or more factors.

• As with other nonparametric tests, ***no assumptions of distributional shape are required***. The test can evaluate one-factor and three-or-more-factor designs but is applied here to the two-factor layout.

• It is more powerful than the two-way ANOVA on ranks method of Conover and Iman (1981) that was for many years the nonparametric test most familiar to scientists for evaluating two factors simultaneously.

• In their simulation study, Brunner and others (1997) found that when all assumptions of ANOVA held, the BDM test had nearly the same power to detect factor effects as did classic ANOVA.

• When there is heteroscedasticity or non-normality, BDM has greater power. It works well even when there are small sample sizes in some combinations of the factors.

• Brunner and others summarize by saying that the test "represents a highly accurate and powerful tool for nonparametric inference in higher-way layouts."

• The null hypothesis of the BDM test is that there are no changes in the cumulative distribution function of data attributable to factor A, factor B, or to an interaction.

• After some interesting yet computationally simple matrix algebra on ranks (in essence, distribution percentiles), a shift in the distribution function attributable to factor A or B will cause an increase in the F-test statistic and rejection of the null hypothesis.

• A shift beyond what is a result of either factor in one or more of the a · b group cells will cause the test for interaction to be significant.

• The procedure is quite analogous to ANOVA, with the difference that ***shifts in group distribution functions are being evaluated rather than shifts in group means***.

Example 2. Iron at low flows—Two-actor BDM test.

```
# Example 2: Iron at low flows—Two-actor BDM test.
data = pd.DataFrame({
    'fe': data_iron.Fe,  # iron concentration
    'mining': data_iron.Mining,
    'rocktype': data_iron.Rocktype
})

# Calculate ranks for the response variable
data['fe_rank'] = rankdata(data['fe'])
```

```
# Fit the OLS model on ranks
model = ols('fe_rank ~ mining * rocktype', data=data).fit()
anova_results = sm.stats.anova_lm(model, typ=2)

# Extract sum of squares and degrees of freedom
SS = anova_results['sum_sq']
df = anova_results['df']

# Calculate mean squares
MS = SS / df

# Calculate the F* statistics
MS_error = MS.iloc[-1]
F_star = MS / MS_error

# Calculate the adjusted degrees of freedom for the BDM test
n = len(data)  # total number of observations
p = len(model.params)  # number of parameters (including intercept)

def adjusted_df(n, df1, df2):
    """Calculate the adjusted degrees of freedom for the BDM test."""
    numerator = (n - df1) * df2
    denominator = df1 + df2 - 1
    return numerator / denominator if denominator != 0 else float('inf')

# Adjusted degrees of freedom for each term
df2_adjusted = [adjusted_df(n, df.iloc[i], df.iloc[-1]) for i in
range(len(df) - 1)]
df2_adjusted.append(df.iloc[-1])  # Add the residual degrees of freedom
without adjustment

# Calculate the P(F > F*)
p_values = [1 - f.cdf(F_star.iloc[i], df.iloc[i], df2_adjusted[i]) for i
in range(len(F_star))]

# Print the results
results = pd.DataFrame({
    'df1': df,
    'df2': df2_adjusted,
    'F*': F_star,
    'P(F > F*)': p_values
})
```

	df1	df2	F*	P(F > F*)
mining	2.0	74.958904	17.740921	4.920162e-07
rocktype	1.0	77.000000	13.375242	4.642748e-04
mining:rocktype	2.0	74.958904	3.709541	2.909338e-02
Residual	72.0	72.000000	1.000000	5.000000e-01

• X1 is the first factor listed in the command (mining)

• X2 is the second (rocktype).

• The interaction term is listed as X1:X2. Although ANOVA was unable to find significance for either factor or for the interaction, all three tests are significant using the nonparametric procedure.

• BDM is a better choice than ANOVA if data violate the normality and constant variance assumptions of ANOVA, and if the objective is to determine whether data values of groups differ from one another.

• If the objective is specifically to determine whether mean values differ between groups, a permutation test is a better choice than ANOVA when ANOVA assumptions are violated.

3 Multiple Comparison Tests

• In most cases the analyst is interested ***not only in whether group medians or means differ***, but ***which groups differ from others***.

• This is information not supplied by the tests presented in the previous sections, but by methods called multiple comparison tests (MCTs).

• MCTs compare (often, all possible) pairs of treatment group medians or means; there are both parametric and nonparametric MCTs.

• With all possible comparisons, interest is in the pattern of group medians or means, such as

mean(group A) = mean(group B) < mean(group C).

• However, if each pair of groups is tested using an $\alpha_{pairwise}$ of 0.05 (the pairwise error rate), the overall probability of making at least one error, called the overall or family error rate, α_{family}, will be much higher than 0.05. With comparisons among all k groups, the number of pairwise comparisons made is $c = \frac{k(k-1)}{2}$ and the family error rate is

$$\alpha_{\text{family}} = 1 - \left(1 - \alpha_{\text{pairwise}}\right)^{c} \qquad \text{Eq. (1)}$$

• For example, when comparing six group means, there are $(6 \cdot 5)/2 = 15(6 \cdot 5)/2 = 15$ pairwise comparisons. If $\alpha_{pairwise} = 0.05$ were used for each test, the probability of making at least one error in the pattern is $\alpha_{family} = 1 - (1 - \alpha pairwise)15 = 0.54$.

• There is about a ***50 percent chance*** that at least one of the comparisons shown in the pattern of the six groups is incorrect.

• MCTs set the α_{family} at the desired level such as 0.05, making the $\alpha_{pariwise}$ much smaller than 0.05. Each individual test must produce a p-value smaller than $\alpha_{pairwise}$ in order for the difference to be significant.

• The much-older Duncan's multiple range, Student-Newman-Keuls (SNK), and Least Significant Difference (LSD) MCTs incorrectly used the pairwise α for each comparison, increasing the probability of finding false differences in the group pattern.

• This led to the recommendation that MCTs should be used only after a significant ANOVA or Kruskal-Wallis test was first found. That recommendation ***is not necessary*** for MCTs using the family error rate, though it is the typical order of testing even now.

• Different MCTs have differing formulae to correct from the family to the pairwise error rate. The simplest is the Bonferroni correction, where $\frac{\alpha_{family}}{c} = \alpha_{pairwise}$.

• For 6 group means, to achieve a Bonferroni family error rate of 0.05, each of the 15 pairwise group comparisons would need a *p*-value below 0.05/15=0.0030.05/15=0.003 to find a significant difference between each pair of group means.

• Most MCT software reports the result of pairwise tests after converting the *p*-value to the family rate equivalent, for this example by multiplying by 15. For example, if one of the 15 tests comparing 2 of the 6 group means achieved an $\alpha_{pairwise}$ of 0.01, it would be reported as $p = 0.15$ on the output so that the user could compare it directly to their α_{family} of 0.05, and know to not reject the null hypothesis.

• If prior to testing it is known that only a few pairwise tests are of interest, for example, sites B, C, and D will be compared only to control site A but not to each other, the number of comparisons made (*c=3*) is fewer than all possible, and α_{family} will be more similar to $\alpha_{pairwise}$. These more specific tests are called contrasts.

• Aho (2016) and Hochberg and Tamhane (1987) review many types of parametric MCTs. Hollander and Wolfe (1999) discuss nonparametric multiple comparisons. Benjamini and Hochberg (1995) developed a ***now widely used correction minimizing the false discovery rate rather than the family error rate***. First developed for genomics, their approach is applicable to water resources, and environmental studies in general.

3.1 Parametric Multiple Comparisons for One-factor ANOVA

• Parametric MCTs compare treatment group means by computing a least significant range (LSR), the distance between any two means that must be exceeded for the two groups to be considered significantly different at a family significance level, α_{family}. If $|\bar{y}_1 - \bar{y}_2| > \text{LSR} = \text{Q}\sqrt{\text{s}^2/n}$, then $\bar{y}_1$ and $\bar{y}_2$ are significantly different.

• The statistic *Q* is analogous to the *t*-statistic in a *t*-test. *Q* depends on the test used and is a function of either a *t*- or studentized range statistic *q*, the error degrees of freedom from the ANOVA, and α_{family}. The variance, s^2, is the mean square for error (residual) from the ANOVA table.

• A few MCTs are valid only for the restrictive case of equal sample sizes within each group. In contrast, ***the Tukey's Honest Significant Difference (HSD), Scheffe, and Bonferroni tests can be used with both equal and unequal group sample sizes***.

• These MCTs compute one least significant range for all pairwise comparisons. The harmonic mean sample size n is substituted for n in the case of unequal group sample sizes.

$$\text{harmonic mean of } n_1 \text{ and } n_2 = \frac{2n_1n_2}{n_1 + n_2}$$

• Of these tests, Tukey's has the most power as its correction from family to pairwise error rates is the smallest. Thus, ***Tukey's has become the standard procedure for parametric MCTs***.

Example 3: Specific capacity—Tukey's multiple comparison test.

The specific capacity data from Knopman (1990) were found to be non-normal and unequal in variance. The natural logs y = ln(specific capacity) better met these two critical assumptions. Boxplots of the natural logs of the data are shown in Figure 4. The ANOVA on logarithms found a significant difference between the four rock types (p = 0.0067). To determine which groups differ from others, Tukey's MCT is now computed.

```
# Example 3: Specific capacity—Tukey's multiple comparison test.
# Path to the RDA file
file_path = './specapic.rda'
# Reading the RDA file
result = pyreadr.read_r(file_path)
data = result['specapic']

df = pd.DataFrame(data)

# Log-transform the specific capacity data
df['log_spcap'] = np.log(df['spcap'])

# Perform ANOVA
model = ols('log_spcap ~ rock', data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)

# Perform Tukey's HSD test
tukey = pairwise_tukeyhsd(endog=df['log_spcap'], groups=df['rock'],
alpha=0.05)
print(tukey)

# Boxplot of the natural logs of the data
plt.figure(figsize=(10, 6))
df.boxplot(column='log_spcap', by='rock', grid=False)
plt.title('Boxplot of Log-transformed Specific Capacity by Rock
Type')
plt.suptitle('')
plt.xlabel('Rock Type')
plt.ylabel('Log(Specific Capacity)')
plt.show()
```

```
              sum_sq     df          F    PR(>F)
rock       54.029544    3.0   4.191556  0.006671
Residual  842.152621  196.0        NaN       NaN
      Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
   group1       group2    meandiff p-adj   lower   upper  reject
-----------------------------------------------------------------
   Dolomite     Limestone  -1.0964 0.0435 -2.1706 -0.0222   True
   Dolomite   Metamorphic  -1.3018 0.0104  -2.376 -0.2275   True
   Dolomite Siliciclastic  -1.1663 0.0275 -2.2406 -0.0921   True
  Limestone   Metamorphic  -0.2053 0.9601 -1.2796  0.8689  False
  Limestone Siliciclastic  -0.0699 0.9983 -1.1441  1.0043  False
Metamorphic Siliciclastic   0.1354 0.9879 -0.9388  1.2097  False
-----------------------------------------------------------------
```

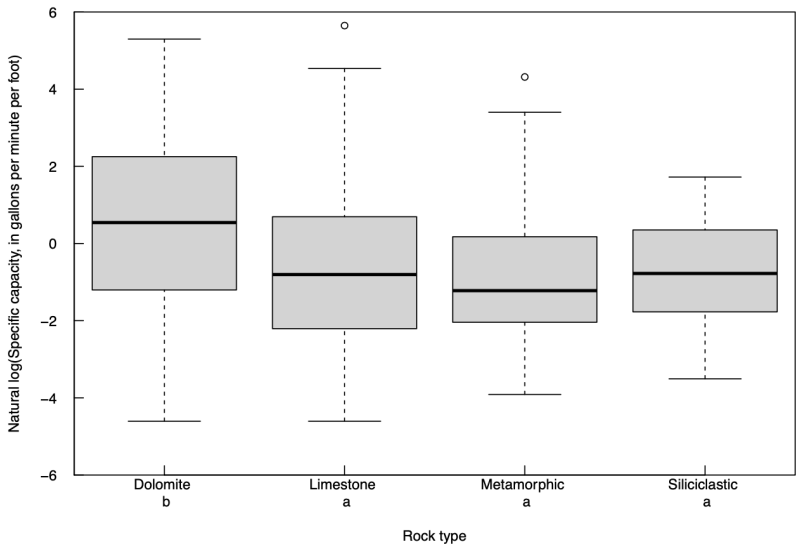


Figure 4. Natural logs of specific capacity of wells in four rock types in Pennsylvania. Letters below rock type names designate group. Data from Knopman (1990).

• Dolomite has a letter “b” which differs from the other groups to show that dolomite has a mean log significantly different from the other groups.

• The other group mean logarithms do not significantly differ from one another, and so receive the same letter “a”.

• Another way to present these results is by plotting the confidence intervals around differences in group means (Figure 5 below).

• If the confidence intervals include zero, there is no significant difference between group means.

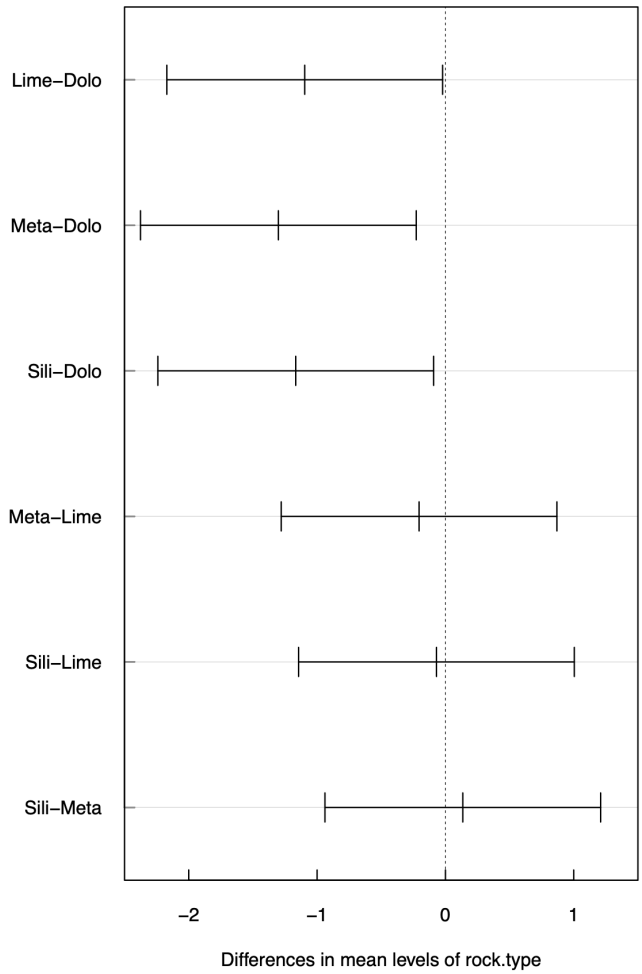


Figure 5. The 95-percent Tukey family confidence intervals on differences in group means (log scale) of the data from Knopman (1990). Lime, limestone; dolo, dolostone; meta, metamorphic; sili, siliciclastic.

3.2 Nonparametric Multiple Comparisons Following the Kruskal-Wallis Test

• A conceptually simple nonparametric MCT to evaluate group patterns following a ***Kruskal-Wallis*** test is to compute all possible pairwise Wilcoxon rank-sum tests, setting the tests' error rates to achieve the family error rate, αfamilyαfamily, of 0.05.

• This MCT creates separate rankings for each test of group pairs, which occasionally could lead to inconsistent results such as A > B, B > C, but A $\not>$ C.

• A strength of using separate ranks is that it allows specific contrasts to be easily computed.

•A ***second common nonparametric*** MCT is Dunn's test (Dunn, 1964). This test differs from pairwise rank-sum tests by using one set of joint (all-group) ranks to test each pairwise difference between groups.

• These are the same ranks used in the ***Kruskal-Wallis*** test. Joint ranking avoids the ***inconsistent*** test result pattern of the pairwise ranking scheme, and has slightly better power to

detect extreme group differences, although the rank-sum approach has slightly better power to detect differences in adjacent groups (Hochberg and Tamhane, 1987).

• Critchlow and Fligner (1991) list several desirable attributes of MCTs, determining that the separate rankings of pairwise rank-sum tests exhibit more of these attributes than does Dunn's test. Hochberg and Tamhane (1987) note that "separate rankings are still generally preferred in practice."

• Of great importance is how ***a family error rate is translated into individual pairwise error rates***. ***Dunn's test may be the most commonly available nonparametric MCT in software***, but it is not the most powerful of the nonparametric MCTs because it uses the Bonferroni correction. The R package PMCMR (Pohlert, 2014) computes the original Dunn's test and a few modifications, including using the BH correction of Benjamini and Hochberg (1995) instead of the Bonferroni correction.

• Benjamini and Hochberg (1995) (BH) developed a correction that controls the false discovery rate, the expected proportion of false positives among the rejected hypotheses.

• The false discovery rate is equivalent to the family error rate when all null hypotheses are true but is smaller otherwise. This criterion is less stringent than the family error rate but is perhaps a more logical objective than α_{family} because it focuses only on comparisons that significantly differ. The resulting benefit is that the BH correction is more powerful than Bonferroni or other methods of adjusting *p*-values.

• Here is how the BH correction works:

1) Suppose we set the false discovery rate, q*, to 0.05. If there are 6 groups there will be $c = 6 \cdot 52c = 26 \cdot 5$ or 15 pairwise comparisons.
2) Compute all 15 comparisons and sort them by their *p*-values from low to high,$i = 1,2, \dots ,15$.
3) Compare each *p*-value to the limit$\frac{i}{c}(q^*)$, starting at i=15, the largest *p*-value.
4) The largest *p*-value is compared to a limit of $\frac{15}{15} \cdot 0.05 = 0.05$.
5) The second largest *p*-value is compared to a limit of $\frac{15}{14} \cdot 0.05 = 0.0467$, and on down to the smallest *p*-value, which is compared to $\frac{15}{1} \cdot 0.05 = 0.0033$.
6) The first computed *p*-value to fall below its limit is considered significant, as are all smaller *p*-values.
7) Suppose this is at the i = 4th lowest *p*-value of 0.010, which had been compared to $\frac{4}{15} \cdot 0.05 = 0.0133$.
8) The pairwise comparisons with the four lowest *p*-values are therefore found to be significant at the false discovery rate of 0.05.
9) In contrast, the Bonferroni correction would have compared all 15 *p*-values to a limit of $\frac{1}{15} \cdot 0.05 = 0.0033$, resulting in fewer significant differences.

• Controlling the false discovery rate is a reasonable goal for Environmental studies.

• It is recommended that the false discovery rate (the BH correction in R) should be the prevalent adjustment method for pairwise rank-sum or other nonparametric MCTs.

Example 4: Specific capacity—Nonparametric multiple comparisons.

The pairwise rank-sum test is computed for the specific capacity data using the BH false-discovery rate *p*-value adjustment:

```
# Example 4: Specific capacity—Nonparametric multiple comparisons.

# Path to the RDA file
file_path = './specapic.rda'
# Reading the RDA file
result = pyreadr.read_r(file_path)
data = result['specapic']

df = pd.DataFrame(data)

# Perform pairwise Wilcoxon rank-sum tests
pairwise_comparisons = []
groups = df['rock'].unique()
for i in range(len(groups)):
    for j in range(i + 1, len(groups)):
        group1 = df[df['rock'] == groups[i]]['spcap']
        group2 = df[df['rock'] == groups[j]]['spcap']
        stat, p_value = ranksums(group1, group2)
        pairwise_comparisons.append((groups[i], groups[j], p_value))

# Adjust p-values using BH method
p_values = [comp[2] for comp in pairwise_comparisons]
_, p_adjusted, _, _ = multipletests(p_values, method='fdr_bh')

# Display results
results = []
for k, (group1, group2, _) in enumerate(pairwise_comparisons):
    results.append({'Comparison': f'{group1} vs {group2}', 'Adjusted
p-value': p_adjusted[k]})

results_df = pd.DataFrame(results)
print(results_df)
```

```
                    Comparison  Adjusted p-value
0         Dolomite vs Limestone          0.042610
1       Dolomite vs Metamorphic          0.012399
2     Dolomite vs Siliciclastic          0.012399
3      Limestone vs Metamorphic          0.802891
4    Limestone vs Siliciclastic          0.893063
5  Metamorphic vs Siliciclastic          0.586106
```

• The *p*-values for each paired comparison in the output have been adjusted to be comparable to the desired α_{family}. For this α_{family} of 0.05, medians for dolomite differ from all three of the other rock types because they are lower than α_{family}.

• No other significant differences are observed. Though the output does not provide a letter diagram as did Tukey's MCT, these test results would match those of the Tukey's letter diagram previously given for the mean logarithms.

• To compute Dunn's test using Python, execute the provided Python code and select the `p_adjust="bonferroni"` option. Do not omit this option, or it will set α (default = 0.05) as the uncorrected $\alpha_{pariwise}$ instead of the α_{family}, an incorrect setting for an MCT.

```
# Perform Kruskal-Wallis test
groups = [df[df['rock'] == group]['spcap'].values for group in
df['rock'].unique()]
stat, p_value = kruskal(*groups)
print(f'Kruskal-Wallis test p-value: {p_value}')

# Perform Dunn's test with Bonferroni correction
dunn_results = sp.posthoc_dunn(df, val_col='spcap', group_col='rock',
p_adjust='bonferroni')

# Suppressing FutureWarnings by modifying the groupby calls in
scikit-posthocs
def custom_posthoc_dunn(x, val_col, group_col, p_adjust=None):
    x = x[[val_col, group_col]].copy()
    x['ranks'] = x[val_col].rank()
    x_lens = x.groupby(group_col, observed=False)[val_col].count()
    x_ranks_avg = x.groupby(group_col,
observed=False)['ranks'].mean()
    # Rest of the function as in the original scikit-posthocs

# Print the results
print(dunn_results)
```

```
Kruskal-Wallis test p-value: 0.009120337563584973
              Dolomite  Limestone  Metamorphic
Limestone     0.065852
Metamorphic   0.011381   1.000000
Siliciclastic 0.072482   1.000000     1.000000
```

• Two-sided p-values for each pairwise comparison are shown. The p-values are adjusted to compare to the α_{family}. Medians for dolomite differ from metamorphic at the 5 percent level.

• No other significant differences are observed. Fewer comparisons are significant than with pairwise rank-sum tests using the BH adjustment. To directly compare the Dunn's test results to pairwise rank-sum tests, use the BH adjustment for Dunn's test as well:

```
# Perform Kruskal-Wallis test with BH adjusted
groups = [df[df['rock'] == group]['spcap'].values for group in
df['rock'].unique()]
stat, p_value = kruskal(*groups)
print(f'Kruskal-Wallis test p-value: {p_value}')

# Perform Dunn's test with Bonferroni correction
```

```
dunn_results = sp.posthoc_dunn(df, val_col='spcap', group_col='rock',
p_adjust='fdr_bh')

# Suppressing FutureWarnings by modifying the groupby calls in
scikit-posthocs
def custom_posthoc_dunn(x, val_col, group_col, p_adjust=None):
    x = x[[val_col, group_col]].copy()
    x['ranks'] = x[val_col].rank()
    x_lens = x.groupby(group_col, observed=False)[val_col].count()
    x_ranks_avg = x.groupby(group_col,
observed=False)['ranks'].mean()
    # Rest of the function as in the original scikit-posthocs

# Print the results
print(dunn_results)
Kruskal-Wallis test p-value: 0.009120337563584973
               Dolomite  Limestone  Metamorphic
Limestone      0.024161
Metamorphic    0.011381   0.688584
Siliciclastic  0.024161   0.973121     0.688584
```

• Using the BH false discovery rate provides ***more power to see pairwise differences*** than the original Bonferroni correction, **as seen by the lower BH *p*-values**.

• The pattern of results with the BH adjustment is the same as with the rank-sum MCT and the Tukey's MCT on logarithms.

• Although there are other nonparametric MCTs, ***the pairwise rank-sum tests with BH correction for family error rate and Dunn's test using the BH correction*** seem to have the most power over a range of conditions.

3.3 Parametric Multiple Comparisons for Two-factor ANOVA

• Tukey's MCT can be computed for ***one of two factors*** in a two-factor ANOVA ***by first adjusting the data for the other factor***.

• If **A** is the factor to be tested, the data are first adjusted for factor **B** by subtracting the mean, δ_j, of the *j*th category of factor **B** from each of the *y* values.

• Remeber that we want to understand their individual and interaction effects on a dependent variable (response). To focus on the effect of factor A alone, we need to remove the influence of factor B from the data. This is done by adjusting the data for factor B.

$$\text{yadj}_{ijk} = y_{ijk} - \delta_j \qquad \text{Eq. (2)}$$

where

δ_j is the influence (mean) of the jth category of factor B.

• Tukey's MCT is then computed on the adjusted y ($yadj_{ijk}$). The error degrees of freedom of the pairwise one-factor tests are also reduced to that of the two-way ANOVA to acknowledge the presence of factor B.

• Tukey's MCT for a two-factor layout requires the same assumptions as the two-factor ANOVA itself—data within each of the a · b treatment groups are required to be ***normally distributed and have equal variance***. Violations of these assumptions will result in a loss of power to detect differences that are actually present.

Example 5: Numerical example to illustrate Tukey's MCT

Example Data:

• Factor A (Treatment): A1,A2,A

• Factor B (Time Points): B1,B2

Response Variable (Growth Rate): Measured for each combination of treatment and time point.

	B1	B2
A1	5.0	7.0
A2	6.0	8.0
A3	7.0	9.0

1. **Calculate Means for Factor B**:
   - Mean growth rate for B1 (δ1): (5.0+6.0+7.0)/3=6.0
   - Mean growth rate for B2 (δ2): (7.0+8.0+9.0)/3=8.0
2. **Adjust Data for Factor B**:
   - Subtract the mean growth rate of each time point from all growth rate observations at that time point.
   - Adjusted data for B1: 5.0−6.0=−1.0, 6.0−6.0=0.0, 7.0−6.0=1.0
   - Adjusted data for B2B2: 7.0−8.0=−1.0, 8.0−8.0=0.0, 9.0−8.0=1.0

	B1	B2
A1	-1.0	-1.0
A2	0.0	0.0
A3	1.0	1.0

Note: Pairwise?

Pairwise Comparisons (Tukey's MCT):

- **Context**: This is used in the context of ANOVA followed by a multiple comparison test.
- **Purpose**: To compare the means of each pair of groups to determine if there are significant differences between them after finding an overall significant effect.
- **Adjustment**: Tukey's HSD (Honestly Significant Difference) adjusts for multiple comparisons to control the family-wise error rate.
- **Procedure**: It involves calculating the difference between each pair of group means and then using the Tukey's HSD test to determine if these differences are statistically significant.
- **Assumptions**: The same as those for ANOVA, including normality and homogeneity of variances within groups.

Pairwise *t*-test:

- **Context**: This is used for comparing the means of two groups at a time, often after an ANOVA if specific group differences are of interest.
- **Purpose**: To test the null hypothesis that the means of two specific groups are equal.
- **Adjustment**: If multiple pairwise *t*-tests are performed, adjustments such as the Bonferroni correction or the Benjamini-Hochberg procedure are used to control for the increased risk of Type I error.
- **Procedure**: It involves performing an ***independent t-test*** for each pair of groups.
- **Types**: It can be either:
  - **Independent *t*-test**: Used when comparing the means of two independent groups.
  - **Paired *t*-test**: Used when comparing means from the same group at different times (repeated measures) or matched pairs.

Differences in Context:

- **Pairwise *t*-test**:
  - Can be performed ***independently without a preceding ANOVA***.
  - Compares only two groups at a time.
  - Requires adjustments if multiple tests are done to control for Type I error.
- **Pairwise Comparisons in Multiple Comparison Tests (like Tukey's MCT)**:
  - Typically performed after a significant ANOVA result.
  - Compares all pairs of groups within a single analysis.
  - Includes built-in adjustments for multiple comparisons to maintain the overall error rate.

Example 6: Iron at low flows—Tukey's multiple comparisons for two-factor ANOVA

• A two-factor ANOVA on the logarithms was previously computed in hte previous example, finding that both factors (mining and rock type) were significant.

• Tukey's MCT is now performed for the primary factor of interest, mining. To focus on this effect only, the name of the primary factor is listed as input to the TukeyHSD function below.

• The function then (internally) subtracts the mean ln(fe) for a rock type from all data of that rock type, to adjust for the ***effect of rock type***.

• It then performs the MCT on the adjusted values by ***mining category***, producing the results.

```
# Example 6: . Iron at low flows—Tukey's multiple comparisons for
two-factor ANOVA.
# Path to the RDA file
file_path = './iron.rda'
# Reading the RDA file
result = pyreadr.read_r(file_path)
data_iron = result['iron']
data_iron['log_Fe'] = np.log(data_iron['Fe'])

print(data_iron)
# Perform two-factor ANOVA
model = ols('log_Fe ~ Mining * Rocktype', data=data_iron).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)

# Perform Tukey's HSD test for the primary factor of interest
(Mining)
tukey_result = pairwise_tukeyhsd(endog=data_iron['log_Fe'],
groups=data_iron['Mining'], alpha=0.05)
print(tukey_result)
```

```
                    sum_sq    df           F    PR(>F)
Mining           69.746910   2.0   15.890508  0.000002
Rocktype         26.312445   1.0   11.989581  0.000904
Mining:Rocktype   2.441812   2.0    0.556320  0.575759
Residual        158.011862  72.0         NaN       NaN
   Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
  group1    group2  meandiff p-adj   lower   upper  reject
-------------------------------------------------------------
Abandoned Reclaimed  -1.2478 0.0154 -2.2943 -0.2013   True
Abandoned   Unmined  -2.3139    0.0 -3.3604 -1.2674   True
Reclaimed   Unmined  -1.0661 0.0449 -2.1127 -0.0196   True
-------------------------------------------------------------
```

From the *p*-values and the direction of differences (meandiff column), the Tukey tests determine that each of the three mining categories have significantly different geometric means, in the order Abandoned > Reclaimed > Unmined.

• There are other Tukey tests for a variety of complicated ANOVA designs available in the TukeyC (Faria and others, 2019) package of R (and Python).

3.4 Nonparametric Multiple Comparisons for Two-factor BDM Test

• A nonparametric MCT analogous to the Tukey's MCT of the previous section for parametric two-factor ANOVA is ***to compute all possible pairwise Wilcoxon rank-sum tests for the primary factor after subtracting medians of y for the secondary factor***.

• Subtraction of medians defined by the second factor adjusts for differences attributable to that factor. This is an adjustment for rocktype in the next example.

Example 7: Iron at low flows—Factorial pairwise rank-sum tests.

• Using Python, first compute a numeric group indicator for rocktype, then compute median iron concentrations for each rocktype, and finally subtract the medians from the iron concentrations to produce the adjusted iron concentrations.

• The relative effects (Rel.effects) indicate the relative levels of adjusted iron concentrations for the three mining categories, in the order Abandoned > Reclaimed > Unmined.

```
# Example 7: Iron at low flows—Factorial pairwise rank-sum tests.
# Path to the RDA file
file_path = './iron.rda'
# Reading the RDA file
result = pyreadr.read_r(file_path)
data_iron = result['iron']

# Compute a numeric group indicator for rocktype
data_iron['Gpind'] = 1 + (data_iron['Rocktype'] ==
'Sandstone').astype(int)

# Compute median iron concentrations for each rocktype
rock_medians = data_iron.groupby('Gpind')['Fe'].median().to_dict()

# Subtract the medians from the iron concentrations to produce the
adjusted iron concentrations
data_iron['feadj'] = data_iron.apply(lambda row: row['Fe'] -
rock_medians[row['Gpind']], axis=1)

# Convert unique mining levels to a list and sort them
mining_levels = sorted(data_iron['Mining'].unique())

# Perform pairwise Wilcoxon rank-sum tests on the adjusted iron
concentrations for the primary factor (Mining)
pairwise_results = []

for i in range(len(mining_levels)):
    for j in range(i + 1, len(mining_levels)):
        group1 = data_iron[data_iron['Mining'] ==
mining_levels[i]]['feadj']
        group2 = data_iron[data_iron['Mining'] ==
mining_levels[j]]['feadj']
        stat, p_value = ranksums(group1, group2)
        pairwise_results.append({
            'Group 1': mining_levels[i],
            'Group 2': mining_levels[j],
            'Statistic': stat,
            'p-value': p_value
        })
```

```
# Display the results of the pairwise Wilcoxon rank-sum tests
pairwise_results_df = pd.DataFrame(pairwise_results)
print(pairwise_results_df)

# Calculate relative effects
data_iron['rank'] = data_iron['feadj'].rank()
relative_effects = data_iron.groupby('Mining')['rank'].mean() /
len(data_iron)
relative_effects =
relative_effects.sort_values(ascending=False).reset_index()
relative_effects.columns = ['Mining', 'Rel.effects']

print(relative_effects)
```

```
     Group 1    Group 2  Statistic   p-value
0  Abandoned  Reclaimed   2.415757  0.015703
1  Abandoned    Unmined   4.117768  0.000038
2  Reclaimed    Unmined   2.937341  0.003310
      Mining  Rel.effects
0  Abandoned     0.682446
1  Reclaimed     0.520464
2    Unmined     0.316321
```

Pairwise Wilcoxon rank-sum tests on the adjusted concentrations using the BH correction for the family error rate show that the three mining categories all significantly differ in their adjusted iron concentrations at $\alpha_{family} = 0.05$.

```
# Perform pairwise Wilcoxon rank-sum tests on the adjusted iron
concentrations for the primary factor (Mining)
pairwise_results = []
p_values = []

for i in range(len(mining_levels)):
    for j in range(i + 1, len(mining_levels)):
        group1 = data_iron[data_iron['Mining'] ==
mining_levels[i]]['feadj']
        group2 = data_iron[data_iron['Mining'] ==
mining_levels[j]]['feadj']
        stat, p_value = ranksums(group1, group2)
        pairwise_results.append({
            'Group 1': mining_levels[i],
            'Group 2': mining_levels[j],
            'Statistic': stat,
            'p-value': p_value
        })
        p_values.append(p_value)

# Adjust p-values for multiple comparisons using the Benjamini-
Hochberg (BH) procedure
reject, pvals_corrected, _, _ = multipletests(p_values,
method='fdr_bh')
```

```
# Add the adjusted p-values to the results
for i in range(len(pairwise_results)):
    pairwise_results[i]['p-adj'] = pvals_corrected[i]
    pairwise_results[i]['reject'] = reject[i]

# Display the results of the pairwise Wilcoxon rank-sum tests with
adjusted p-values
pairwise_results_df = pd.DataFrame(pairwise_results)
print(pairwise_results_df)

# Calculate relative effects
data_iron['rank'] = data_iron['feadj'].rank()
relative_effects = data_iron.groupby('Mining')['rank'].mean() /
len(data_iron)
relative_effects =
relative_effects.sort_values(ascending=False).reset_index()
relative_effects.columns = ['Mining', 'Rel.effects']

print(relative_effects)
```

```
     Group 1    Group 2  Statistic   p-value     p-adj  reject
0  Abandoned  Reclaimed   2.415757  0.015703  0.015703    True
1  Abandoned    Unmined   4.117768  0.000038  0.000115    True
2  Reclaimed    Unmined   2.937341  0.003310  0.004966    True
      Mining  Rel.effects
0  Abandoned     0.682446
1  Reclaimed     0.520464
2    Unmined     0.316321
```