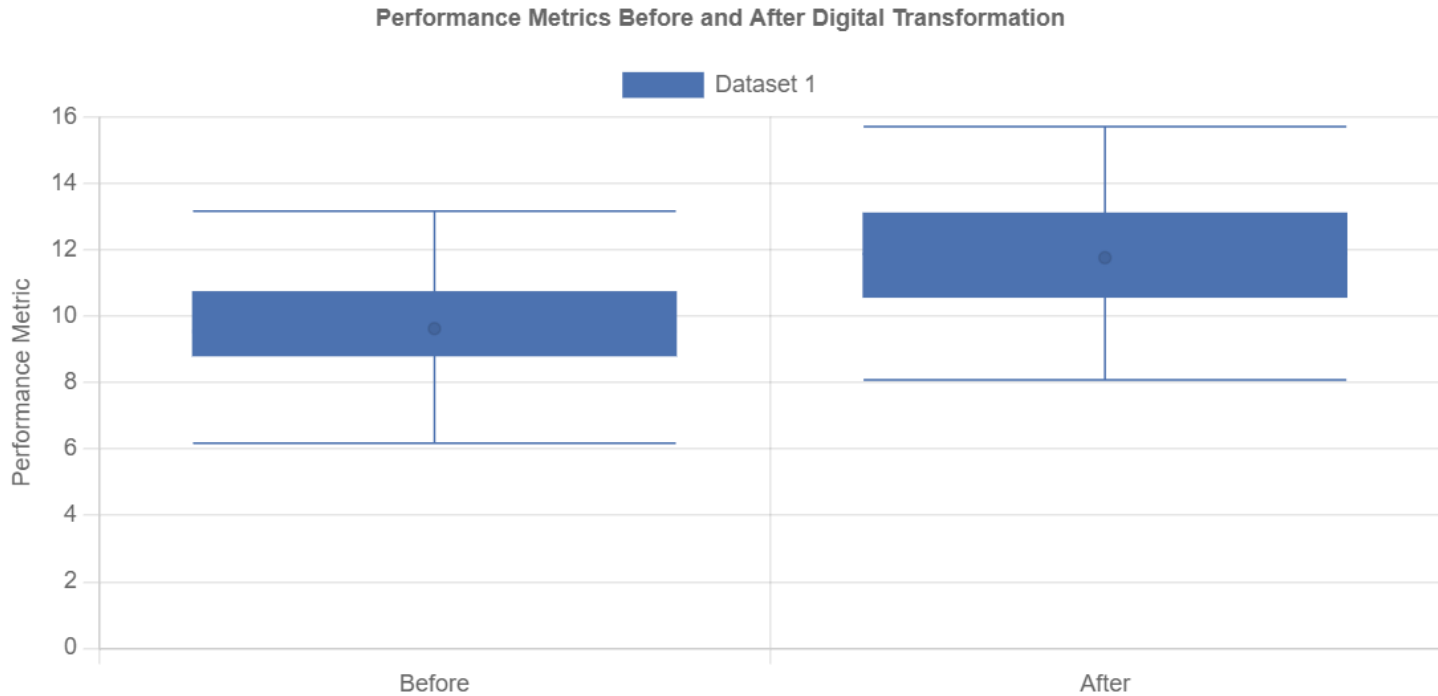
Appendix

Leeds Doctoral College - University of Leeds
Quantitative Data Analysis
T-Test: One-sample t-test , Two-sample t-test, Paired t-test , and Welch's t-test in Python
https://colab.research.google.com/drive/1-IGnUbP8F3gWTFzap1s-ZEiTuUI_fY98
By Heider Jeffer

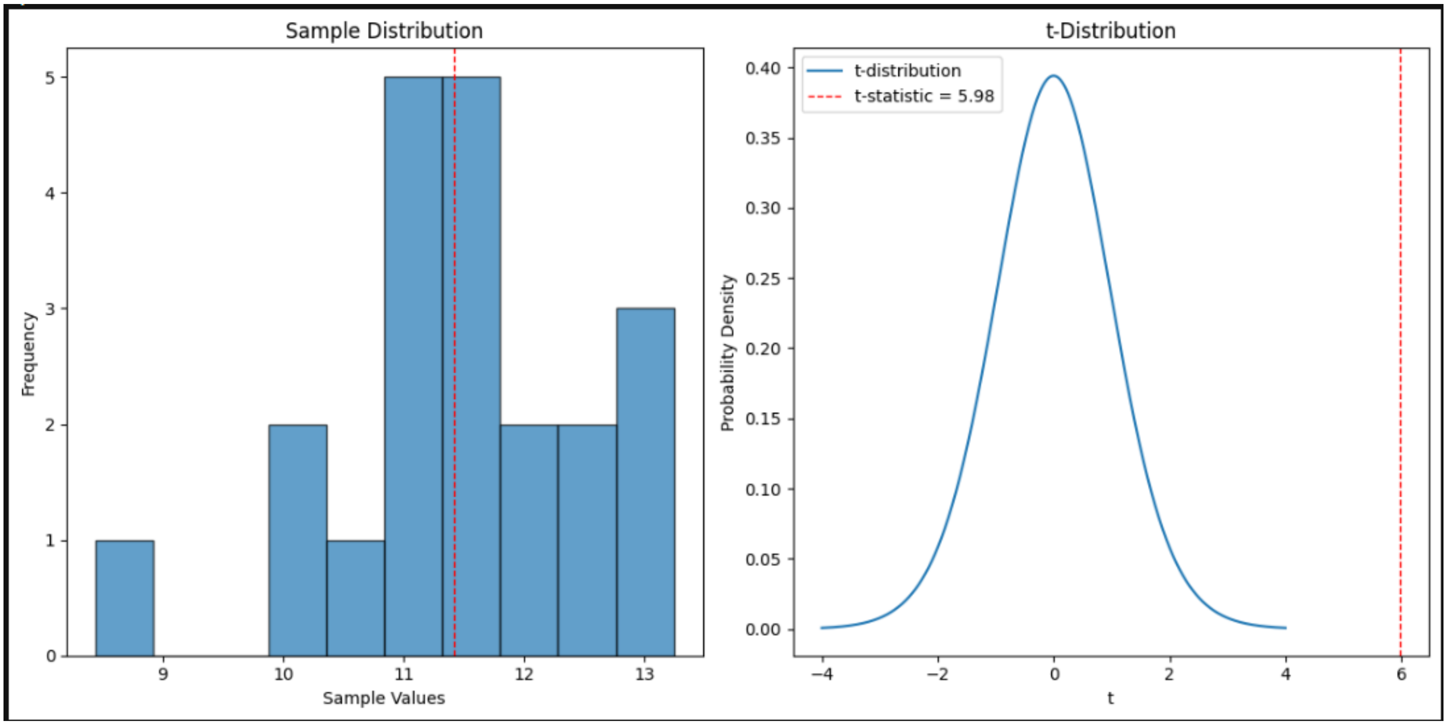



```
# Leeds Doctoral College - University of Leeds
# Quantitative Data Analysis with Python
# By Heider Jeffer
# June 20, 2024

# One-sample t-test example in Python

import numpy as np
from scipy import stats

# Population Mean
mu = 10

# Sample Size
N1 = 21

# Degrees of freedom
dof = N1 - 1

# Generate a random sample with mean = 11 and standard deviation = 1
x = np.random.randn(N1) + 11

# Using the Stats library, compute t-statistic and p-value
t_stat, p_val = stats.ttest_1samp(a=x, popmean = mu)
print("t-statistic = " + str(t_stat))
print("p-value = " + str(p_val))
```

```
t-statistic = 3.2806238272741637
p-value = 0.0037389287836883793
```

```python
# Plot One-sample t-test example in Python by Heider Jeffer
# Compute the t-statistic and p-value,
# and then plot the sample distribution
# along with the t-distribution for visualization

import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Population Mean
mu = 10

# Sample Size
N1 = 21

# Generate a random sample with mean = 11 and standard deviation = 1
np.random.seed(0)  # Set seed for reproducibility
x = np.random.randn(N1) + 11

# Using the Stats library, compute t-statistic and p-value
t_stat, p_val = stats.ttest_1samp(a=x, popmean=mu)
print("t-statistic = " + str(t_stat))
print("p-value = " + str(p_val))

# Plot the sample distribution and t-distribution
plt.figure(figsize=(12, 6))

# Plot sample distribution
plt.subplot(1, 2, 1)
plt.hist(x, bins=10, edgecolor='black', alpha=0.7)
plt.axvline(np.mean(x), color='r', linestyle='dashed', linewidth=1)
plt.title('Sample Distribution')
plt.xlabel('Sample Values')
plt.ylabel('Frequency')

# Plot t-distribution with sample t-statistic
plt.subplot(1, 2, 2)
x_vals = np.linspace(-4, 4, 1000)
t_dist = stats.t.pdf(x_vals, df=N1-1)
plt.plot(x_vals, t_dist, label='t-distribution')
plt.axvline(t_stat, color='r', linestyle='dashed', linewidth=1, label=f't-statistic = {t_stat:.2f}')
plt.title('t-Distribution')
plt.xlabel('t')
plt.ylabel('Probability Density')
plt.legend()

plt.tight_layout()
plt.show()
```

```
t-statistic = 5.9753587422612915
p-value = 7.646525308171602e-06
```

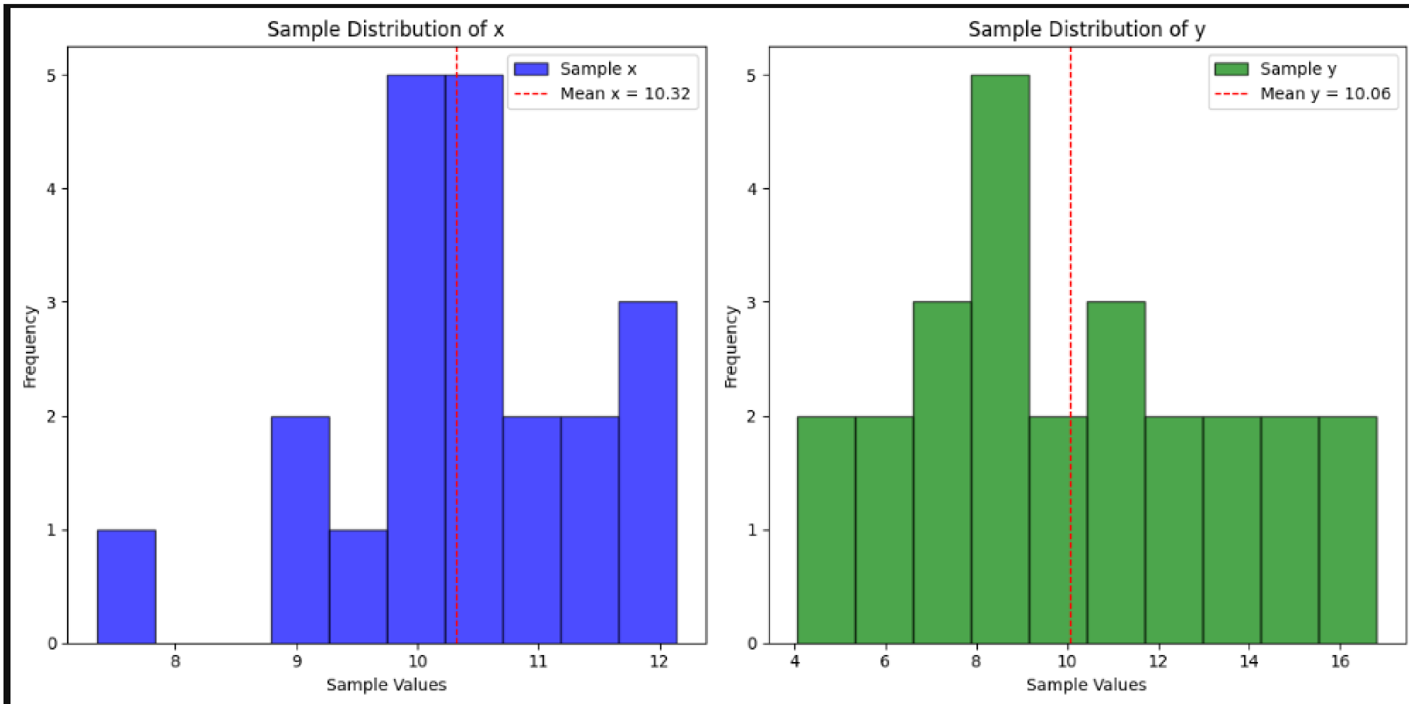



```
A different approach
1. compute the sample mean (x_bar)
2. ompute the sample standard deviation with the degree of freedom of one (it represents the standard deviation of the sample
3. Compute the standard error
4. Use the one-sample t-statistic formula above
5. Compute the p-value to establish the significance of the t-statistic.


Sample Mean
bar = x.mean()

Standard Deviation
d = np.std(x, ddof=1)

Standard Error
e = std/np.sqrt(N1)

Calculating the T-Statistics
stat = (x_bar - mu) / ste

p-value of the t-statistic
val = 2*(1 - stats.t.cdf(abs(t_stat), df = dof))
int("t-statistic = " + str(t_stat))
int("p-value = " + str(p_val))
```

```
t-statistic = 3.8068345338354854
p-value = 0.0011047641393842067
```

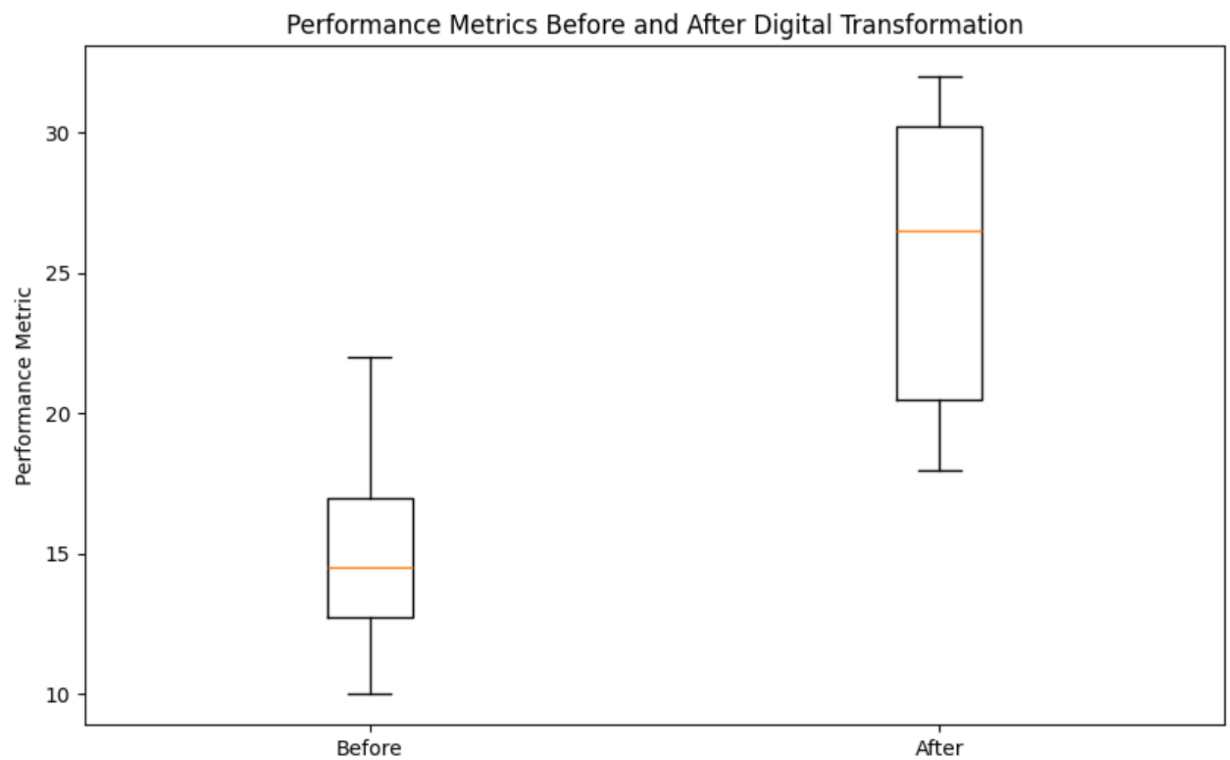
```python
# Calculate the Welch's t-test and plot the distributions of the two samples to generate the plots
# Run this code in Python  to visualize the distributions of the two samples.
# This will help you see the overlap and understand
# why the p-value is high, indicating no significant difference between the means.
# by Heider Jeffer

import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Sample Sizes
N1, N2 = 21, 25

# Degrees of freedom
dof = min(N1, N2) - 1

# Gaussian distributed data with mean = 9.9 and var = 1
np.random.seed(0)  # Set seed for reproducibility
x = np.random.randn(N1) + 9.9

# Gaussian distributed data with mean = 10 and var = 3
y = 3 * np.random.randn(N2) + 10

# Using SciPy Package
t_stat, p_val = stats.ttest_ind(x, y, equal_var=False)
print("t-statistic = " + str(t_stat))
print("p-value = " + str(p_val))

# Plot the sample distributions
plt.figure(figsize=(12, 6))

# Plot distribution of x
plt.subplot(1, 2, 1)
plt.hist(x, bins=10, edgecolor='black', alpha=0.7, color='blue', label='Sample x')
plt.axvline(np.mean(x), color='r', linestyle='dashed', linewidth=1, label=f'Mean x = {np.mean(x):.2f}')
plt.title('Sample Distribution of x')
plt.xlabel('Sample Values')
plt.ylabel('Frequency')
plt.legend()

# Plot distribution of y
plt.subplot(1, 2, 2)
plt.hist(y, bins=10, edgecolor='black', alpha=0.7, color='green', label='Sample y')
plt.axvline(np.mean(y), color='r', linestyle='dashed', linewidth=1, label=f'Mean y = {np.mean(y):.2f}')
plt.title('Sample Distribution of y')
plt.xlabel('Sample Values')
plt.ylabel('Frequency')
plt.legend()

plt.tight_layout()
plt.show()
```

```
t-statistic = 0.3562853342360152
p-value = 0.7241634784025108
```




```
# Given a typical significance level (α) of 0.05, a p-value of 0.724.
# 0.724 is much greater than 0.05. This means:
# 1. There is weak evidence against the null hypothesis.
# 2. We fail to reject the null hypothesis.
# 3. There is not a statistically significant difference between the means of the two samples.
```

Leeds Doctoral College - University of Leeds
Quantitative Data Analysis
ANOVA in Python https://colab.research.google.com/drive/1-Q672tb7HZAU_u7HoVtsZ4plsrDS8QU-
By Heider Jeffer



In [1]:

```python
# Leeds Doctoral College - University of Leeds
# Quantitative Data Analysis with Python
# ANOVA
# in Python
# By Heider Jeffer
# June 20, 2024


import pandas as pd
from scipy.stats import f_oneway

# Sample data creation (if not reading from a file)
data = {
    'metric': [10, 12, 15, 14, 13, 16, 20, 22, 21, 19, 18, 24, 30, 32, 29, 31],  # Sample metrics
    'period': ['before', 'before', 'before', 'before', 'before', 'before', 'before', 'before',
               'after', 'after', 'after', 'after', 'after', 'after', 'after', 'after']  # 'before' or 'after'
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Separate the data into two groups
before = df[df['period'] == 'before']['metric']
after = df[df['period'] == 'after']['metric']

# Perform ANOVA
f_statistic, p_value = f_oneway(before, after)

# Output the results
print(f"F-statistic: {f_statistic}")
print(f"P-value: {p_value}")

# Interpretation of the results
alpha = 0.05
if p_value < alpha:
    print("There is a significant difference between the performance metrics before and after digital transformation.")
else:
    print("There is no significant difference between the performance metrics before and after digital transformation.")
```

```
F-statistic: 17.329896907216494
P-value: 0.000957171784251362
There is a significant difference between the performance metrics before and after digital transformation.
```

:

```python
# Plot Welch's t-test by Heider Jeffer
import pandas as pd
from scipy.stats import f_oneway
import matplotlib.pyplot as plt

# Sample data creation (if not reading from a file)
data = {
    'metric': [10, 12, 15, 14, 13, 16, 20, 22, 21, 19, 18, 24, 30, 32, 29, 31],  # Sample metrics
    'period': ['before', 'before', 'before', 'before', 'before', 'before', 'before', 'before',
               'after', 'after', 'after', 'after', 'after', 'after', 'after', 'after']  # 'before' or 'after'
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Separate the data into two groups
before = df[df['period'] == 'before']['metric']
after = df[df['period'] == 'after']['metric']

# Perform ANOVA
f_statistic, p_value = f_oneway(before, after)

# Output the results
print(f"F-statistic: {f_statistic}")
print(f"P-value: {p_value}")

# Interpretation of the results
alpha = 0.05
if p_value < alpha:
    print("There is a significant difference between the performance metrics before and after digital transformation.")
else:
    print("There is no significant difference between the performance metrics before and after digital transformation.")

# Plotting the results
plt.figure(figsize=(10, 6))

# Create box plots
plt.boxplot([before, after], tick_labels=['Before', 'After'])

# Add titles and labels
plt.title('Performance Metrics Before and After Digital Transformation')
plt.ylabel('Performance Metric')

# Show plot
plt.show()
```

```
F-statistic: 17.329896907216494
P-value: 0.000957171784251362
There is a significant difference between the performance metrics before and after digital transformation.
```