Analysis script · PY

```
#!/usr/bin/env python3
"""
Statistical Analysis Script
Study: Between Tradition and Algorithm: Human-AI Co-Creation in Textile Design
Education
Journal: Humanities and Social Sciences Communications
Submission ID: 64cfd2aa-cbcf-443c-accd-b4adfbb3a28d

Requirements: Python 3.11+, pandas, scipy, numpy
Usage: python Analysis_Script.py
"""

import pandas as pd
import numpy as np
from scipy import stats

# ── 1. LOAD DATA ──────────────────────────────────────────────
df = pd.read_excel('Kilim_Data_Anonymised.xlsx', sheet_name='Raw_Ratings')

criteria = [
'Originality', 'Aesthetics', 'Cultural_Fidelity',
'Commercial_Viability', 'Creativity',
'Technical_Feasibility', 'Compositional_Integrity'
]

human = df[df['Condition'] == 'Human']
ai_assisted = df[df['Condition'] == 'AI-Assisted']
full_ai = df[df['Condition'] == 'Full AI']

# ── 2. DESCRIPTIVE STATISTICS ─────────────────────────────────
print("=" * 65)
print("DESCRIPTIVE STATISTICS BY CONDITION")
print("=" * 65)
print(f"{'Criterion':<28} {'Human':>10} {'AI-Assist':>10} {'Full AI':>10}")
print("-" * 65)
for c in criteria:
h_m, h_s = human[c].mean(), human[c].std()
a_m, a_s = ai_assisted[c].mean(), ai_assisted[c].std()
f_m, f_s = full_ai[c].mean(), full_ai[c].std()
print(f"{c:<28} {h_m:.2f}({h_s:.2f}) {a_m:.2f}({a_s:.2f}) {f_m:.2f}({f_s:.2f})")

# ── 3. KRUSKAL-WALLIS TESTS ───────────────────────────────────
print("\n" + "=" * 65)
print("KRUSKAL-WALLIS H TESTS (omnibus)")
print("=" * 65)
print(f"{'Criterion':<28} {'H':>8} {'p':>10} {'Sig':>6}")
print("-" * 65)
kw_results = {}
for c in criteria:
g1 = human[c].dropna()
g2 = ai_assisted[c].dropna()
g3 = full_ai[c].dropna()
H, p = stats.kruskal(g1, g2, g3)
sig = "***" if p < 0.001 else "**" if p < 0.01 else "*" if p < 0.05 else "n.s."
print(f"{c:<28} {H:>8.2f} {p:>10.4f} {sig:>6}")
kw_results[c] = (H, p)

# ── 4. POST-HOC MANN-WHITNEY U (Bonferroni alpha=0.0167) ──────
print("\n" + "=" * 65)
print("POST-HOC MANN-WHITNEY U TESTS (Bonferroni corrected, alpha=.0167)")
print("=" * 65)
ALPHA = 0.0167
comparisons = [
('Human', human),
('AI-Assisted', ai_assisted),
('Full AI', full_ai)
```

```
]

for c in criteria:
print(f"\n{c}:")
for i, (n1, g1) in enumerate(comparisons):
for n2, g2 in comparisons[i+1:]:
d1 = g1[c].dropna()
d2 = g2[c].dropna()
U, p = stats.mannwhitneyu(d1, d2, alternative='two-sided')
sig = "***" if p < 0.001 else "**" if p < 0.01 else "*" if p < ALPHA else "n.s."
bonf = "significant" if p < ALPHA else "not significant"
print(f" {n1} vs {n2}: U={U:.0f}, p={p:.4f} {sig} ({bonf})")

# ── 5. EVALUATOR GROUP COMPARISON ──────────────────────────────
print("\n" + "=" * 65)
print("EVALUATOR GROUP: Academics vs Industry Designers")
print("=" * 65)
academics = df[df['Evaluator_Group'] == 'Academic']
industry = df[df['Evaluator_Group'] == 'Industry']
print(f"{'Criterion':<28} {'Academic':>10} {'Industry':>10} {'p':>10} {'Sig':>6}")
print("-" * 65)
for c in criteria:
d1 = academics[c].dropna()
d2 = industry[c].dropna()
U, p = stats.mannwhitneyu(d1, d2, alternative='two-sided')
sig = "***" if p < 0.001 else "**" if p < 0.01 else "*" if p < 0.05 else "n.s."
print(f"{c:<28} {d1.mean():>10.2f} {d2.mean():>10.2f} {p:>10.4f} {sig:>6}")

print("\nAnalysis complete.")
```