Manuel de Préparation: Tests Non-Paramétriques

Introduction

Les tests non-paramétriques (ou tests distribution-free) sont des alternatives aux tests paramétriques classiques lorsque les hypothèses de normalité ne sont pas respectées. Dans le contexte bancaire, où les distributions sont souvent asymétriques (revenus, montants de transactions), ces tests sont particulièrement utiles.

Partie 1: Quand Utiliser les Tests Non-Paramétriques?

1.1 Conditions d'Utilisation

Utiliser les tests non-paramétriques quand: - La distribution n'est pas normale - L'échantillon est petit ($n < 30$) - Les données sont ordinales (pas quantitatives) - Il y a des outliers importants - Les variances sont très inégales

Avantages: - Moins d'hypothèses requises - Robustes aux outliers - Applicables aux données ordinales

Inconvénients: - Moins de puissance statistique (si normalité respectée) - Moins informatifs (basés sur les rangs)

1.2 Correspondance Tests Paramétriques vs Non-Paramétriques

Situation	Test Paramétrique	Test Non-Paramétrique
2 groupes indépendants	t-test indépendant	**Mann-Whitney U**
2 groupes appariés	t-test apparié	**Wilcoxon signé**
3+ groupes indépendants	ANOVA	**Kruskal-Wallis**
3+ groupes appariés	ANOVA mesures répétées	**Friedman**
Corrélation	Pearson	**Spearman** / **Kendall**
Association catégorielle	-	**Chi-carré**

1.3 Mnémotechnique: "MWKF-SK"

```
M - Mann-Whitney: 2 groupes indépendants
W - Wilcoxon: 2 groupes appariés
K - Kruskal-Wallis: 3+ groupes indépendants
F - Friedman: 3+ groupes appariés
S - Spearman: Corrélation de rang
K - Kendall: Corrélation de rang (alternative)
```

Partie 2: Test de Mann-Whitney U

2.1 Description

Alternative au: t-test pour échantillons indépendants

Hypothèses: - H_0: Les deux groupes ont la même distribution - H_1: Les distributions sont différentes

Principe: Compare les rangs des observations entre les deux groupes.

2.2 Implémentation Python

```python
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt

# Exemple bancaire: Montants de transactions par type de client
np.random.seed(42)

# Clients retail (distribution asymétrique)
montants_retail = np.random.exponential(scale=5000, size=50)

# Clients premium (montants plus élevés, aussi asymétriques)
montants_premium = np.random.exponential(scale=15000, size=45)

# Visualiser les distributions
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].hist(montants_retail, bins=20, alpha=0.7, label='Retail')
axes[0].hist(montants_premium, bins=20, alpha=0.7, label='Premium')
axes[0].legend()
axes[0].set_title('Distribution des Montants')

# Test de normalité
_, p_retail = stats.shapiro(montants_retail)
_, p_premium = stats.shapiro(montants_premium)
print(f"Shapiro-Wilk Retail: p = {p_retail:.4f}")
print(f"Shapiro-Wilk Premium: p = {p_premium:.4f}")
# Si p < 0.05, distribution non normale → utiliser Mann-Whitney

# Test de Mann-Whitney U
statistic, p_value = stats.mannwhitneyu(
    montants_retail,
    montants_premium,
    alternative='two-sided'
)

print(f"\n=== Test de Mann-Whitney U ===")
print(f"Statistique U: {statistic:.2f}")
print(f"P-value: {p_value:.4f}")

if p_value < 0.05:
    print("→ Différence significative entre les groupes")
else:
    print("→ Pas de différence significative")

# Taille d'effet (r = Z / √N)
n1, n2 = len(montants_retail), len(montants_premium)
```

```
z_score = stats.norm.ppf(1 - p_value/2)
effect_size = z_score / np.sqrt(n1 + n2)
print(f"Taille d'effet (r): {effect_size:.3f}")
```

2.3 Interprétation de la Statistique U

```
U mesure le nombre de fois où une observation d'un groupe
dépasse une observation de l'autre groupe.

U minimum = 0 (aucun chevauchement)
U maximum = n₁ × n₂ (chevauchement total)

Taille d'effet (r):
- r < 0.3: Petit effet
- r = 0.3-0.5: Effet moyen
- r > 0.5: Grand effet
```

2.4 Application Bancaire

```
"""
Cas: Comparer les délais de remboursement entre deux agences
"""

# Données: Jours de retard de paiement
agence_a = [5, 12, 3, 45, 7, 2, 8, 15, 90, 4, 6, 10, 25, 3, 8]
agence_b = [2, 4, 1, 5, 3, 2, 6, 4, 3, 2, 5, 4, 7, 3, 2, 4, 5]

# Test
stat, p = stats.mannwhitneyu(agence_a, agence_b, alternative='greater')

print("=== Comparaison des Délais de Retard ===")
print(f"Médiane Agence A: {np.median(agence_a):.1f} jours")
print(f"Médiane Agence B: {np.median(agence_b):.1f} jours")
print(f"P-value (A > B): {p:.4f}")

if p < 0.05:
    print("\n⚠ L'Agence A a des retards significativement plus longs")
    print("   Recommandation: Audit des processus de recouvrement")
```

Partie 3: Test de Wilcoxon Signé

3.1 Description

Alternative au: t-test pour échantillons appariés

Usage: Comparer deux mesures sur les mêmes sujets (avant/après, deux conditions)

Hypothèses: - H_0: La distribution des différences est symétrique autour de 0 - H_1: Les différences ne sont pas centrées sur 0

3.2 Implémentation Python

```python
from scipy.stats import wilcoxon

# Exemple: Satisfaction client avant et après une campagne
np.random.seed(42)

n_clients = 30
satisfaction_avant = np.random.randint(1, 8, n_clients)  # Score 1-10
# Amélioration après campagne (pas toujours)
satisfaction_apres = satisfaction_avant + np.random.choice([-1, 0, 1, 2], n_clients, p=[0.
satisfaction_apres = np.clip(satisfaction_apres, 1, 10)

# Calculer les différences
differences = satisfaction_apres - satisfaction_avant

print("=== Analyse Avant/Après Campagne ===")
print(f"Satisfaction moyenne avant: {satisfaction_avant.mean():.2f}")
print(f"Satisfaction moyenne après: {satisfaction_apres.mean():.2f}")
print(f"Différence médiane: {np.median(differences):.2f}")

# Test de Wilcoxon
statistic, p_value = wilcoxon(satisfaction_avant, satisfaction_apres, alternative='less')

print(f"\nTest de Wilcoxon:")
print(f"Statistique W: {statistic:.2f}")
print(f"P-value: {p_value:.4f}")

if p_value < 0.05:
    print("\n✓ Amélioration significative de la satisfaction")
else:
    print("\n✗ Pas d'amélioration significative")
```

3.3 Application Bancaire

```python
"""
Cas: Évaluer l'effet d'une formation sur la productivité des conseillers
"""

# Productivité (nombre de dossiers traités par jour)
productivite_avant = [12, 15, 8, 10, 14, 9, 11, 13, 7, 16]
productivite_apres = [14, 17, 10, 12, 15, 11, 14, 16, 9, 18]

# Test
stat, p = wilcoxon(productivite_avant, productivite_apres, alternative='less')

print("=== Évaluation de la Formation ===")
print(f"Productivité médiane avant: {np.median(productivite_avant):.1f}")
print(f"Productivité médiane après: {np.median(productivite_apres):.1f}")
print(f"P-value: {p:.4f}")

if p < 0.05:
    amelioration = np.median(productivite_apres) - np.median(productivite_avant)
```

```
    print(f"\n✓ Formation efficace (+{amelioration:.1f} dossiers/jour)")
    print("   ROI: Calculer coût formation vs gain productivité")
```

Partie 4: Test de Kruskal-Wallis

4.1 Description

Alternative à: ANOVA à un facteur

Usage: Comparer 3 groupes ou plus

Hypothèses: - H_0: Toutes les populations ont la même distribution - H_1: Au moins une population diffère

---

4.2 Implémentation Python

```
from scipy.stats import kruskal

# Exemple: Temps de traitement de dossiers par agence
np.random.seed(42)

agence_centre = np.random.exponential(5, 25)  # Jours
agence_nord = np.random.exponential(7, 30)
agence_sud = np.random.exponential(4, 28)

# Test de Kruskal-Wallis
stat, p = kruskal(agence_centre, agence_nord, agence_sud)

print("=== Comparaison des Temps de Traitement ===")
print(f"Médiane Centre: {np.median(agence_centre):.2f} jours")
print(f"Médiane Nord: {np.median(agence_nord):.2f} jours")
print(f"Médiane Sud: {np.median(agence_sud):.2f} jours")
print(f"\nKruskal-Wallis H: {stat:.2f}")
print(f"P-value: {p:.4f}")

if p < 0.05:
    print("\n⚠ Différences significatives entre agences")
    print("   → Effectuer des tests post-hoc")
```

4.3 Tests Post-Hoc (Comparaisons Multiples)

```
from scipy.stats import mannwhitneyu
from itertools import combinations

def posthoc_mannwhitney(groups, group_names, alpha=0.05):
    """Tests post-hoc Mann-Whitney avec correction de Bonferroni"""

    n_comparisons = len(list(combinations(range(len(groups)), 2)))
    alpha_corrected = alpha / n_comparisons
```

```
    print(f"\n=== Tests Post-Hoc (Bonferroni α = {alpha_corrected:.4f}) ===")

    results = []
    for (i, j) in combinations(range(len(groups)), 2):
        stat, p = mannwhitneyu(groups[i], groups[j], alternative='two-sided')
        significant = "***" if p < alpha_corrected else "n.s."
        print(f"{group_names[i]} vs {group_names[j]}: U={stat:.1f}, p={p:.4f} {significant
        results.append({
            'Comparaison': f"{group_names[i]} vs {group_names[j]}",
            'U': stat,
            'p-value': p,
            'Significatif': p < alpha_corrected
        })

    return pd.DataFrame(results)

# Application
groups = [agence_centre, agence_nord, agence_sud]
names = ['Centre', 'Nord', 'Sud']
posthoc_mannwhitney(groups, names)
```

4.4 Application Bancaire

```
"""
Cas: Comparer les scores de satisfaction par segment client
"""

# Scores de satisfaction (1-10)
retail = [6, 5, 7, 4, 6, 5, 8, 6, 5, 7, 4, 6, 5, 7, 6]
premium = [7, 8, 9, 7, 8, 9, 8, 7, 9, 8, 7, 8]
private = [9, 10, 8, 9, 10, 9, 9, 10, 8, 9]

# Test
stat, p = kruskal(retail, premium, private)

print("=== Satisfaction par Segment ===")
print(f"Médiane Retail: {np.median(retail):.1f}")
print(f"Médiane Premium: {np.median(premium):.1f}")
print(f"Médiane Private: {np.median(private):.1f}")
print(f"\nKruskal-Wallis: H={stat:.2f}, p={p:.4f}")

if p < 0.05:
    print("\n✓ Différences significatives de satisfaction")
    print("   Priorité: Améliorer l'expérience Retail")
```

Partie 5: Test de Friedman

5.1 Description

Alternative à: ANOVA à mesures répétées

Usage: Comparer 3+ conditions sur les mêmes sujets

5.2 Implémentation Python

```python
from scipy.stats import friedmanchisquare

# Exemple: Évaluation de 3 processus par les mêmes évaluateurs
np.random.seed(42)

n_evaluateurs = 12
processus_a = np.random.randint(5, 10, n_evaluateurs)
processus_b = np.random.randint(6, 10, n_evaluateurs)
processus_c = np.random.randint(4, 9, n_evaluateurs)

# Test de Friedman
stat, p = friedmanchisquare(processus_a, processus_b, processus_c)

print("=== Comparaison des Processus ===")
print(f"Score médian Processus A: {np.median(processus_a):.1f}")
print(f"Score médian Processus B: {np.median(processus_b):.1f}")
print(f"Score médian Processus C: {np.median(processus_c):.1f}")
print(f"\nFriedman χ²: {stat:.2f}")
print(f"P-value: {p:.4f}")

if p < 0.05:
    print("\n✓ Différences significatives entre processus")
```

Partie 6: Corrélations Non-Paramétriques

6.1 Corrélation de Spearman

Usage: Corrélation basée sur les rangs (relation monotone, pas nécessairement linéaire)

```python
from scipy.stats import spearmanr

# Exemple: Relation ancienneté et fidélité client
anciennete = [2, 5, 1, 8, 3, 10, 4, 6, 12, 7]
score_fidelite = [45, 72, 30, 88, 55, 95, 60, 78, 92, 80]

# Corrélation de Spearman
rho, p = spearmanr(anciennete, score_fidelite)

print("=== Corrélation Ancienneté-Fidélité ===")
print(f"Spearman ρ: {rho:.3f}")
print(f"P-value: {p:.4f}")

# Interprétation
if abs(rho) < 0.3:
    force = "faible"
elif abs(rho) < 0.7:
    force = "modérée"
```

```
else:
    force = "forte"

direction = "positive" if rho > 0 else "négative"
print(f"\n→ Corrélation {force} et {direction}")
```

6.2 Corrélation de Kendall (Tau)

Usage: Alternative à Spearman, plus robuste pour petits échantillons

```
from scipy.stats import kendalltau

tau, p = kendalltau(anciennete, score_fidelite)

print("=== Corrélation de Kendall ===")
print(f"Kendall τ: {tau:.3f}")
print(f"P-value: {p:.4f}")
```

6.3 Comparaison Pearson vs Spearman vs Kendall

Critère	Pearson	Spearman	Kendall
Type de relation	Linéaire	Monotone	Monotone
Type de données	Continues	Ordinales/Continues	Ordinales/Continues
Sensibilité outliers	Élevée	Faible	Faible
Petits échantillons	Moins fiable	OK	Meilleur
Interprétation	Variance partagée	Concordance des rangs	Paires concordantes

---

6.4 Application Bancaire: Choix de la Corrélation

```
"""
Cas: Analyser la relation entre revenus et montant de dépôts
"""
import matplotlib.pyplot as plt

# Données (revenus très asymétriques)
np.random.seed(42)
revenus = np.random.exponential(50000, 100)  # Distribution asymétrique
depots = 0.3 * revenus + np.random.exponential(10000, 100)

# Visualiser
fig, ax = plt.subplots(figsize=(8, 6))
ax.scatter(revenus, depots, alpha=0.6)
ax.set_xlabel('Revenus (HTG)')
ax.set_ylabel('Dépôts (HTG)')
ax.set_title('Relation Revenus-Dépôts')

# Calculer les trois corrélations
```

```python
from scipy.stats import pearsonr, spearmanr, kendalltau

r_pearson, p_pearson = pearsonr(revenus, depots)
r_spearman, p_spearman = spearmanr(revenus, depots)
r_kendall, p_kendall = kendalltau(revenus, depots)

print("=== Comparaison des Corrélations ===")
print(f"Pearson r:  {r_pearson:.3f} (p={p_pearson:.4f})")
print(f"Spearman ρ: {r_spearman:.3f} (p={p_spearman:.4f})")
print(f"Kendall τ:  {r_kendall:.3f} (p={p_kendall:.4f})")

# Recommandation
print("\n Recommandation:")
print("   Distribution asymétrique → Utiliser Spearman")
print(f"   Interprétation: Corrélation modérée positive ({r_spearman:.2f})")
```

Partie 7: Test du Chi-Carré

7.1 Chi-Carré d'Indépendance

Usage: Tester l'association entre deux variables catégorielles

```python
from scipy.stats import chi2_contingency

# Exemple: Type de client vs type de produit préféré
# Tableau de contingence
data = np.array([
    [120, 80, 50],   # Retail: Épargne, Crédit, Assurance
    [60, 100, 40],   # Premium
    [20, 30, 50]     # Private
])

chi2, p, dof, expected = chi2_contingency(data)

print("=== Test du Chi-Carré d'Indépendance ===")
print(f"Chi² = {chi2:.2f}")
print(f"Degrés de liberté = {dof}")
print(f"P-value = {p:.4f}")

if p < 0.05:
    print("\n✓ Association significative entre segment et produit préféré")
else:
    print("\n✗ Pas d'association significative")

# Tableau des fréquences attendues
print("\nFréquences attendues:")
print(pd.DataFrame(expected,
                   index=['Retail', 'Premium', 'Private'],
                   columns=['Épargne', 'Crédit', 'Assurance']).round(1))

# V de Cramer (taille d'effet)
n = data.sum()
min_dim = min(data.shape) - 1
```

```python
v_cramer = np.sqrt(chi2 / (n * min_dim))
print(f"\nV de Cramer: {v_cramer:.3f}")
```

7.2 Chi-Carré de Conformité (Goodness of Fit)

```python
from scipy.stats import chisquare

# Exemple: Vérifier si les défauts sont uniformément répartis par trimestre
defauts_observes = [45, 52, 38, 65]  # Par trimestre
defauts_attendus = [50, 50, 50, 50]  # Si uniforme

chi2, p = chisquare(defauts_observes, defauts_attendus)

print("=== Test de Conformité ===")
print(f"Défauts observés: {defauts_observes}")
print(f"Défauts attendus: {defauts_attendus}")
print(f"\nChi² = {chi2:.2f}, p = {p:.4f}")

if p < 0.05:
    print("\n⚠ Distribution non uniforme")
    print("   Le Q4 a significativement plus de défauts")
```

Partie 8: Tableau Récapitulatif des Tests

8.1 Guide de Sélection

```
┌─────────────────────────────────────────────────────────────┐
│                    SÉLECTION DU TEST                        │
├─────────────────────────────────────────────────────────────┤
│  Données normales?                                          │
│  ├── OUI → Tests paramétriques                              │
│  │    ├── 2 groupes indép. → t-test                         │
│  │    ├── 2 groupes appariés → t-test apparié               │
│  │    ├── 3+ groupes indép. → ANOVA                         │
│  │    └── Corrélation → Pearson                             │
│  │                                                          │
│  └── NON → Tests non-paramétriques                          │
│       ├── 2 groupes indép. → Mann-Whitney U                 │
│       ├── 2 groupes appariés → Wilcoxon signé               │
│       ├── 3+ groupes indép. → Kruskal-Wallis                │
│       ├── 3+ groupes appariés → Friedman                    │
│       ├── Corrélation → Spearman / Kendall                  │
│       └── Variables catégorielles → Chi-carré               │
└─────────────────────────────────────────────────────────────┘
```

8.2 Formules Clés

Test	Statistique	Distribution
Mann-Whitney	$U = n_1n_2 + n_1(n_1+1)/2 - R_1$	Approx. normale si n > 20
Wilcoxon	$W = \Sigma$ rangs positifs	Approx. normale si n > 25
Kruskal-Wallis	$H = 12/N(N+1) \times \Sigma(R^2_i/n_i) - 3(N+1)$	Chi-carré (k-1)
Spearman	$\rho = 1 - 6\Sigma d^2_i / n(n^2-1)$	t avec n-2 df
Chi-carré	$\chi^2 = \Sigma(O-E)^2/E$	Chi-carré (df)

Partie 9: Code Express

9.1 Fonction de Sélection Automatique

```python
def choisir_test_comparaison(groupe1, groupe2, apparie=False, alpha=0.05):
    """
    Choisit et exécute le test approprié pour comparer deux groupes
    """
    from scipy import stats

    print("=== ANALYSE COMPARATIVE ===\n")

    # Étape 1: Tester la normalité
    _, p1 = stats.shapiro(groupe1) if len(groupe1) < 5000 else (0, 0)
    _, p2 = stats.shapiro(groupe2) if len(groupe2) < 5000 else (0, 0)

    normal1 = p1 > alpha
    normal2 = p2 > alpha

    print(f"Normalité Groupe 1: {'Oui' if normal1 else 'Non'} (p={p1:.4f})")
    print(f"Normalité Groupe 2: {'Oui' if normal2 else 'Non'} (p={p2:.4f})")

    # Étape 2: Choisir le test
    if normal1 and normal2:
        print("\n→ Distributions normales: Test paramétrique")
        if apparie:
            stat, p = stats.ttest_rel(groupe1, groupe2)
            test_nom = "t-test apparié"
        else:
            stat, p = stats.ttest_ind(groupe1, groupe2)
            test_nom = "t-test indépendant"
    else:
        print("\n→ Distributions non normales: Test non-paramétrique")
        if apparie:
            stat, p = stats.wilcoxon(groupe1, groupe2)
            test_nom = "Wilcoxon signé"
        else:
            stat, p = stats.mannwhitneyu(groupe1, groupe2)
            test_nom = "Mann-Whitney U"

    # Résultats
    print(f"\nTest utilisé: {test_nom}")
    print(f"Statistique: {stat:.4f}")
    print(f"P-value: {p:.4f}")
```

```
    if p < alpha:
        print(f"\n✓ SIGNIFICATIF (p < {alpha})")
    else:
        print(f"\n✗ NON SIGNIFICATIF (p ≥ {alpha})")

    return {'test': test_nom, 'statistic': stat, 'p_value': p}

# Utilisation
result = choisir_test_comparaison(agence_a, agence_b)
```

9.2 Rapport Complet Non-Paramétrique

```
def rapport_nonparametrique(groupes, noms, alpha=0.05):
    """
    Génère un rapport complet pour l'analyse non-paramétrique
    """
    from scipy import stats

    print("=" * 60)
    print("RAPPORT D'ANALYSE NON-PARAMÉTRIQUE")
    print("=" * 60)

    # Statistiques descriptives
    print("\n1. STATISTIQUES DESCRIPTIVES")
    print("-" * 40)
    for nom, groupe in zip(noms, groupes):
        print(f"\n{nom}:")
        print(f"  N = {len(groupe)}")
        print(f"  Médiane = {np.median(groupe):.2f}")
        print(f"  IQR = {np.percentile(groupe, 75) - np.percentile(groupe, 25):.2f}")
        print(f"  Min-Max = [{np.min(groupe):.2f}, {np.max(groupe):.2f}]")

    # Tests de normalité
    print("\n2. TESTS DE NORMALITÉ (Shapiro-Wilk)")
    print("-" * 40)
    for nom, groupe in zip(noms, groupes):
        if len(groupe) < 5000:
            stat, p = stats.shapiro(groupe)
            result = "Normal" if p > alpha else "Non normal"
            print(f"{nom}: W={stat:.4f}, p={p:.4f} → {result}")

    # Test global
    print("\n3. TEST GLOBAL (Kruskal-Wallis)")
    print("-" * 40)
    if len(groupes) >= 2:
        stat, p = stats.kruskal(*groupes)
        print(f"H = {stat:.4f}")
        print(f"P-value = {p:.4f}")
        if p < alpha:
            print("→ Différences significatives détectées")
```

```
            # Tests post-hoc
            print("\n4. TESTS POST-HOC (Mann-Whitney avec Bonferroni)")
            print("-" * 40)
            from itertools import combinations
            n_comp = len(list(combinations(range(len(groupes)), 2)))
            alpha_adj = alpha / n_comp

            for (i, j) in combinations(range(len(groupes)), 2):
                stat_mw, p_mw = stats.mannwhitneyu(groupes[i], groupes[j])
                sig = "***" if p_mw < alpha_adj else "n.s."
                print(f"{noms[i]} vs {noms[j]}: U={stat_mw:.1f}, p={p_mw:.4f} {sig}")
        else:
            print("→ Pas de différences significatives")

    print("\n" + "=" * 60)

# Utilisation
rapport_nonparametrique(
    [agence_centre, agence_nord, agence_sud],
    ['Centre', 'Nord', 'Sud']
)
```

Résumé Express: Questions Probables

1. **"Quand utiliser Mann-Whitney au lieu du t-test?"** → Quand la normalité n'est pas respectée ou données ordinales
2. **"Quelle est la différence entre Spearman et Pearson?"** → Pearson: relation linéaire, Spearman: relation monotone (basé sur les rangs)
3. **"Comment interpréter le test de Kruskal-Wallis?"** → Alternative à l'ANOVA, si $p < 0.05$ → au moins un groupe diffère
4. **"Quand utiliser le test de Wilcoxon?"** → Données appariées (avant/après) non normales
5. **"Qu'est-ce que le V de Cramer?"** → Mesure de taille d'effet pour le Chi-carré (0 = pas d'association, 1 = association parfaite)

Checklist Tests Non-Paramétriques

```
□ Vérifier la normalité (Shapiro-Wilk)
□ Si p < 0.05 ou données ordinales → non-paramétrique
□ Identifier le design:
  □ 2 groupes indép. → Mann-Whitney
  □ 2 groupes appariés → Wilcoxon
  □ 3+ groupes indép. → Kruskal-Wallis + post-hoc
  □ 3+ groupes appariés → Friedman + post-hoc
  □ Corrélation → Spearman
  □ Association catégorielle → Chi-carré
□ Calculer la taille d'effet
□ Appliquer correction de Bonferroni si comparaisons multiples
```

□ Interpréter dans le contexte bancaire

Document préparé pour l'examen Data Analyst - UniBank Haiti Tests Non-Paramétriques: L'alternative robuste