Manuel de Préparation: Statistiques Inférentielles - Enquêtes et Tests d'Hypothèses

Introduction

Les statistiques inférentielles permettent de tirer des conclusions sur une population à partir d'un échantillon. Dans le contexte bancaire, elles sont essentielles pour valider des hypothèses business, mesurer l'efficacité des campagnes, et prendre des décisions basées sur les données.

Partie 1: Fondements de l'Inférence Statistique

1.1 Population vs Échantillon

```
Population (N): Ensemble complet de tous les éléments d'intérêt
Échantillon (n): Sous-ensemble de la population

Exemple bancaire:
Population: Tous les clients de la banque (500,000)
Échantillon: 1,000 clients sélectionnés pour une enquête
```

Pourquoi Échantillonner?

- Coût et temps réduits
- Impossibilité pratique d'étudier toute la population
- Tests destructifs (ex: audits approfondis)

Paramètres vs Statistiques

Concept	Population	Échantillon
Moyenne	μ (mu)	$\bar{x}$ (x-bar)
Écart-type	σ (sigma)	s
Proportion	π ou p	$\hat{p}$ (p-hat)

1.2 Types d'Échantillonnage

Échantillonnage Probabiliste

Méthode	Description	Usage
Aléatoire simple	Chaque élément a même probabilité	Base, populations homogènes
Stratifié	Division en strates, puis échantillonnage	Populations hétérogènes
Par grappes	Sélection de groupes entiers	Réduction des coûts
Systématique	Sélection à intervalle régulier	Listes ordonnées

Contexte Bancaire

```python
# Échantillonnage stratifié par segment client
import pandas as pd

def stratified_sample(df, strata_col, n_per_strata):
    """Échantillon stratifié"""
    return df.groupby(strata_col, group_keys=False).apply(
        lambda x: x.sample(min(len(x), n_per_strata))
    )

# Exemple: 100 clients par segment
sample = stratified_sample(df_clients, 'segment', 100)
```

1.3 Distribution d'Échantillonnage

Théorème Central Limite (TCL)

```
Pour n suffisamment grand (n  30), la distribution des moyennes
d'échantillons tend vers une distribution normale, quelle que soit
la distribution de la population.

Moyenne:  x̄ =
Écart-type:  x̄ =   / √n  (erreur standard)
```

Implications

- Permet d'utiliser la distribution normale pour l'inférence
- Plus n est grand, plus l'erreur standard est faible
- Justifie les intervalles de confiance et tests d'hypothèses

---

Partie 2: Intervalles de Confiance

2.1 Concept

Définition: Plage de valeurs dans laquelle on est "confiant" que le paramètre de population se trouve.

```
IC = Estimation ponctuelle ± Marge d'erreur

IC pour la moyenne:
IC = x̄ ± z( /2) × (s/√n)
```

2.2 Niveaux de Confiance

Niveau	z(α/2)	Interprétation
90%	1.645	90% de chances que μ soit dans l'intervalle
95%	1.96	Standard en recherche
99%	2.576	Très conservateur

2.3 Calcul en Python

```python
from scipy import stats
import numpy as np

def confidence_interval_mean(data, confidence=0.95):
    """Calcule l'intervalle de confiance pour la moyenne"""
    n = len(data)
    mean = np.mean(data)
    se = stats.sem(data)  # Standard error

    # Pour grands échantillons (z)
    z = stats.norm.ppf((1 + confidence) / 2)
    margin = z * se

    # Pour petits échantillons (t)
    # t = stats.t.ppf((1 + confidence) / 2, n-1)
    # margin = t * se

    return mean - margin, mean + margin

# Exemple
data = df['montant_transaction'].values
lower, upper = confidence_interval_mean(data, 0.95)
print(f"IC 95%: [{lower:.2f}, {upper:.2f}]")
```

2.4 Intervalle de Confiance pour une Proportion

```python
def confidence_interval_proportion(successes, n, confidence=0.95):
    """IC pour une proportion"""
    p_hat = successes / n
    z = stats.norm.ppf((1 + confidence) / 2)
    se = np.sqrt(p_hat * (1 - p_hat) / n)
    margin = z * se

    return p_hat - margin, p_hat + margin

# Exemple: Taux de satisfaction
n_satisfaits = 450
n_total = 500
lower, upper = confidence_interval_proportion(n_satisfaits, n_total, 0.95)
print(f"Taux de satisfaction IC 95%: [{lower:.1%}, {upper:.1%}]")
```

2.5 Taille d'Échantillon Requise

```python
def sample_size_mean(margin_error, std_dev, confidence=0.95):
    """Taille d'échantillon pour estimer une moyenne"""
    z = stats.norm.ppf((1 + confidence) / 2)
    n = (z * std_dev / margin_error) ** 2
    return int(np.ceil(n))

def sample_size_proportion(margin_error, p_estimate=0.5, confidence=0.95):
    """Taille d'échantillon pour estimer une proportion"""
    z = stats.norm.ppf((1 + confidence) / 2)
```

```
    n = (z ** 2 * p_estimate * (1 - p_estimate)) / (margin_error ** 2)
    return int(np.ceil(n))

# Exemple: Enquête de satisfaction avec marge de 3%
n_required = sample_size_proportion(0.03, 0.5, 0.95)
print(f"Échantillon requis: {n_required}")  # ~1068
```

Partie 3: Tests d'Hypothèses

3.1 Cadre Conceptuel

Hypothèses

```
H  (Hypothèse nulle): Pas d'effet, pas de différence
H  (Hypothèse alternative): Il y a un effet ou une différence

Exemple:
H : Le taux de conversion n'a pas changé (  =  )
H : Le taux de conversion a changé (    )
```

Types de Tests

Type	H_1	Usage
Bilatéral	$\mu \neq \mu_0$	Différence dans n'importe quelle direction
Unilatéral droit	$\mu > \mu_0$	Augmentation attendue
Unilatéral gauche	$\mu < \mu_0$	Diminution attendue

3.2 Erreurs et Risques

	H_0 vraie	H_0 fausse
Rejeter H_0	Erreur Type I (α)	Décision correcte
Ne pas rejeter H_0	Décision correcte	Erreur Type II (β)

```
  (alpha): Risque de faux positif (typiquement 0.05)
  (beta): Risque de faux négatif
Puissance = 1 -  : Probabilité de détecter un vrai effet
```

3.3 La p-value

Définition: Probabilité d'observer un résultat aussi extrême que celui observé, si H_0 est vraie.

```
Si p-value   : Rejeter H  (résultat statistiquement significatif)
Si p-value >  : Ne pas rejeter H

Attention:
- p-value faible   effet important
- Significativité statistique   Significativité pratique
```

3.4 Tests Paramétriques

Test t pour un échantillon (One-sample t-test) Hypothèses:

H : =
H : (ou > ou <)

Conditions: - Données continues - Échantillon aléatoire - Distribution approximativement normale (ou n > 30)

```
from scipy.stats import ttest_1samp

# Exemple: Le solde moyen est-il différent de 10,000 HTG?
data = df['solde'].values
t_stat, p_value = ttest_1samp(data, 10000)

alpha = 0.05
if p_value < alpha:
    print(f"Rejeter H : Le solde moyen diffère de 10,000 (p={p_value:.4f})")
else:
    print(f"Ne pas rejeter H  (p={p_value:.4f})")
```

Test t pour deux échantillons indépendants Hypothèses:

H : =
H :

```
from scipy.stats import ttest_ind

# Exemple: Les soldes diffèrent-ils entre segments?
group1 = df[df['segment'] == 'Premium']['solde']
group2 = df[df['segment'] == 'Standard']['solde']

t_stat, p_value = ttest_ind(group1, group2)
print(f"t-statistic: {t_stat:.3f}, p-value: {p_value:.4f}")
```

Test t pour échantillons appariés (Paired t-test) **Usage:** Mesures avant/après sur les mêmes sujets

```
from scipy.stats import ttest_rel

# Exemple: La formation a-t-elle amélioré les ventes?
avant = df['ventes_avant']
apres = df['ventes_apres']

t_stat, p_value = ttest_rel(apres, avant)
print(f"Amélioration significative? p-value: {p_value:.4f}")
```

3.5 Test Z pour les Proportions

Une proportion

```
from statsmodels.stats.proportion import proportions_ztest

# Exemple: Le taux de défaut est-il supérieur à 5%?
```

```python
count = 60  # Nombre de défauts
nobs = 1000  # Total de prêts
p0 = 0.05  # Proportion hypothétique

z_stat, p_value = proportions_ztest(count, nobs, p0, alternative='larger')
print(f"z-statistic: {z_stat:.3f}, p-value: {p_value:.4f}")
```

Deux proportions

```python
# Exemple: Les taux de conversion diffèrent-ils entre campagnes?
counts = np.array([120, 100])  # Succès
nobs = np.array([500, 500])    # Totaux

z_stat, p_value = proportions_ztest(counts, nobs)
print(f"Différence significative? p-value: {p_value:.4f}")
```

3.6 Test du Chi-Carré (χ^2)

Test d'indépendance Hypothèses:

H : Les variables sont indépendantes
H : Les variables sont dépendantes

```python
from scipy.stats import chi2_contingency

# Tableau de contingence
contingency_table = pd.crosstab(df['segment'], df['produit_achete'])

chi2, p_value, dof, expected = chi2_contingency(contingency_table)

print(f"Chi²: {chi2:.2f}, p-value: {p_value:.4f}")
print(f"Degrés de liberté: {dof}")
```

Exemple Bancaire

```python
# Les préférences de canal diffèrent-elles selon l'âge?
contingency = pd.crosstab(df['tranche_age'], df['canal_prefere'])
chi2, p, dof, expected = chi2_contingency(contingency)

if p < 0.05:
    print("Les préférences de canal dépendent de l'âge")
```

Test d'ajustement (Goodness of fit)

```python
from scipy.stats import chisquare

# Les transactions sont-elles uniformément réparties par jour?
observed = df['jour_semaine'].value_counts().sort_index().values
expected = [len(df) / 7] * 7  # Distribution uniforme

chi2_stat, p_value = chisquare(observed, expected)
```

3.7 ANOVA (Analysis of Variance)

ANOVA à un facteur Hypothèses:

H : = = ... = (toutes les moyennes égales)
H : Au moins une moyenne diffère

```
from scipy.stats import f_oneway

# Exemple: Les soldes moyens diffèrent-ils entre agences?
groups = [group['solde'].values for name, group in df.groupby('agence')]
f_stat, p_value = f_oneway(*groups)

print(f"F-statistic: {f_stat:.3f}, p-value: {p_value:.4f}")
```

Post-hoc: Tukey HSD

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Après ANOVA significative, quelles paires diffèrent?
tukey = pairwise_tukeyhsd(df['solde'], df['agence'], alpha=0.05)
print(tukey.summary())
```

3.8 Tests Non-Paramétriques

Quand les utiliser?

- Données ordinales
- Distribution non normale
- Petits échantillons
- Présence d'outliers

Tests Équivalents

Paramétrique	Non-paramétrique	Usage
t-test 1 échantillon	Wilcoxon signed-rank	Comparer à une valeur
t-test 2 échantillons	Mann-Whitney U	Comparer 2 groupes
t-test apparié	Wilcoxon signed-rank	Avant/après
ANOVA	Kruskal-Wallis	Comparer 3+ groupes
Chi-carré	Fisher exact	Petits effectifs

```
from scipy.stats import mannwhitneyu, wilcoxon, kruskal

# Mann-Whitney U
u_stat, p_value = mannwhitneyu(group1, group2)

# Kruskal-Wallis
h_stat, p_value = kruskal(*groups)
```

Partie 4: Corrélation et Association

4.1 Corrélation de Pearson

Hypothèses: - Relation linéaire - Variables continues - Distribution bivariée normale

```
from scipy.stats import pearsonr

r, p_value = pearsonr(df['revenu'], df['montant_pret'])
print(f"Corrélation: {r:.3f}, p-value: {p_value:.4f}")
```

Interprétation

```
|r| < 0.3: Corrélation faible
0.3  |r| < 0.7: Corrélation modérée
|r|  0.7: Corrélation forte
```

4.2 Corrélation de Spearman

Pour données ordinales ou relations non linéaires.

```
from scipy.stats import spearmanr

rho, p_value = spearmanr(df['satisfaction'], df['anciennete'])
```

4.3 Coefficient V de Cramer

Pour deux variables catégorielles.

```
from scipy.stats import chi2_contingency

def cramers_v(contingency_table):
    chi2 = chi2_contingency(contingency_table)[0]
    n = contingency_table.sum().sum()
    min_dim = min(contingency_table.shape) - 1
    return np.sqrt(chi2 / (n * min_dim))

v = cramers_v(pd.crosstab(df['segment'], df['produit']))
print(f"V de Cramer: {v:.3f}")
```

Partie 5: Enquêtes et Sondages

5.1 Conception d'Enquête

Étapes

```
1. Définir les objectifs
2. Identifier la population cible
3. Déterminer la méthode d'échantillonnage
4. Calculer la taille d'échantillon
5. Concevoir le questionnaire
6. Pré-tester
7. Collecter les données
8. Analyser et rapporter
```

Types de Questions

Type	Avantages	Inconvénients
Fermées	Faciles à analyser	Options limitées
Ouvertes	Richesse d'information	Difficiles à coder
Likert	Mesure d'attitude	Biais de tendance centrale
Dichotomiques	Simple	Perte de nuance

5.2 Biais à Éviter

Biais	Description	Prévention
Non-réponse	Certains ne répondent pas	Relances, incitations
Auto-sélection	Répondants non représentatifs	Échantillonnage aléatoire
Acquiescement	Tendance à dire "oui"	Questions inversées
Désirabilité sociale	Réponses "acceptables"	Questions indirectes
Effet d'ordre	Ordre des questions influence	Randomisation

5.3 Analyse des Résultats d'Enquête

```python
def analyze_survey(df, question_col, group_col=None):
    """Analyse une question d'enquête"""

    results = {}

    # Distribution des réponses
    results['distribution'] = df[question_col].value_counts(normalize=True)

    # Score moyen (si Likert)
    if df[question_col].dtype in ['int64', 'float64']:
        results['mean'] = df[question_col].mean()
        results['std'] = df[question_col].std()
        results['ci'] = confidence_interval_mean(df[question_col].values)

    # Par groupe si spécifié
    if group_col:
        results['by_group'] = df.groupby(group_col)[question_col].agg(['mean', 'std', 'count'])

        # Test de différence
        groups = [group[question_col].values for _, group in df.groupby(group_col)]
        if len(groups) == 2:
            _, p_value = ttest_ind(*groups)
        else:
            _, p_value = f_oneway(*groups)
        results['p_value'] = p_value

    return results
```

Partie 6: Applications Bancaires

6.1 Test A/B pour Campagnes Marketing

```
def ab_test_conversion(control, treatment, alpha=0.05):
    """
    Test A/B pour taux de conversion
    control, treatment: tuples (succès, total)
    """
    c_success, c_total = control
    t_success, t_total = treatment

    c_rate = c_success / c_total
    t_rate = t_success / t_total

    # Test de proportion
    counts = np.array([c_success, t_success])
    nobs = np.array([c_total, t_total])
    z_stat, p_value = proportions_ztest(counts, nobs)

    # Lift
    lift = (t_rate - c_rate) / c_rate * 100

    return {
        'control_rate': c_rate,
        'treatment_rate': t_rate,
        'lift': lift,
        'p_value': p_value,
        'significant': p_value < alpha
    }

# Exemple
results = ab_test_conversion(
    control=(100, 1000),    # 10% conversion
    treatment=(130, 1000)   # 13% conversion
)
```

6.2 Analyse de Satisfaction Client (NPS)

```
def analyze_nps(scores):
    """Analyse du Net Promoter Score"""

    promoters = (scores >= 9).sum()
    passives = ((scores >= 7) & (scores < 9)).sum()
    detractors = (scores < 7).sum()
    total = len(scores)

    nps = (promoters - detractors) / total * 100

    # IC pour le NPS
    # Approximation: traiter comme différence de proportions
    se = np.sqrt(
        (promoters/total * (1 - promoters/total) +
         detractors/total * (1 - detractors/total)) / total
```

```
    )
    margin = 1.96 * se * 100

    return {
        'nps': nps,
        'ci_lower': nps - margin,
        'ci_upper': nps + margin,
        'promoters_pct': promoters / total * 100,
        'passives_pct': passives / total * 100,
        'detractors_pct': detractors / total * 100
    }
```

6.3 Test de Différence de Taux de Défaut

```
def compare_default_rates(segment1, segment2):
    """Compare les taux de défaut entre deux segments"""

    # segment: dict with 'defaults' and 'total'
    rate1 = segment1['defaults'] / segment1['total']
    rate2 = segment2['defaults'] / segment2['total']

    counts = np.array([segment1['defaults'], segment2['defaults']])
    nobs = np.array([segment1['total'], segment2['total']])

    z_stat, p_value = proportions_ztest(counts, nobs)

    return {
        'rate_segment1': rate1,
        'rate_segment2': rate2,
        'difference': rate1 - rate2,
        'z_statistic': z_stat,
        'p_value': p_value
    }
```

Partie 7: Résumé des Tests

Choisir le Bon Test

```
Question: Comparer une moyenne à une valeur?
→ t-test un échantillon (ou Wilcoxon si non-normal)

Question: Comparer deux moyennes?
→ t-test indépendant (Mann-Whitney si non-normal)

Question: Comparer avant/après?
→ t-test apparié (Wilcoxon si non-normal)

Question: Comparer 3+ moyennes?
→ ANOVA (Kruskal-Wallis si non-normal)

Question: Tester une proportion?
→ Test z de proportion
```

```
Question: Comparer deux proportions?
→ Test z deux proportions ou Chi-carré

Question: Variables catégorielles liées?
→ Chi-carré d'indépendance

Question: Corrélation entre variables continues?
→ Pearson (Spearman si non-linéaire)
```

Questions d'Entretien Fréquentes

1. **Qu'est-ce qu'une p-value et comment l'interpréter?** → Probabilité d'observer le résultat si H_0 est vraie; $p < \alpha$ signifie significatif
2. **Différence entre erreur Type I et Type II?** → Type I = faux positif (rejeter H_0 vraie); Type II = faux négatif (ne pas rejeter H_0 fausse)
3. **Quand utiliser un test paramétrique vs non-paramétrique?** → Paramétrique si distribution normale et grandes données; non-paramétrique sinon
4. **Comment déterminer la taille d'échantillon?** → Formule basée sur marge d'erreur souhaitée, niveau de confiance, et variabilité
5. **Qu'est-ce que la puissance statistique?** → Probabilité de détecter un vrai effet (1-β); typiquement 80%

Checklist Tests d'Hypothèses

```
Définir clairement H  et H
Choisir le niveau de signification
Vérifier les conditions d'application du test
Sélectionner le test approprié
Calculer la statistique de test et p-value
Prendre la décision statistique
Interpréter dans le contexte business
Considérer la taille de l'effet (pas juste p-value)
Mentionner les limites de l'analyse
```

Rappel final: La significativité statistique n'est pas tout. Toujours considérer: - La significativité pratique (taille de l'effet) - Le contexte business - Les implications des erreurs (Type I vs Type II)