A/B Testing et Expérimentation - Guide Complet pour Data Analyst Bancaire

Introduction

L'A/B testing (ou test contrôlé randomisé) est une méthode expérimentale essentielle pour valider des hypothèses business avec des données. Dans le contexte bancaire, il permet de mesurer l'impact réel des changements avant leur déploiement à grande échelle.

Mnémotechnique: HIPPO - Ne laissez pas le **H**ighest **I**ncome **P**erson's **P**ersonal **O**pinion décider. Laissez les données trancher!

Partie 1: Fondements de l'A/B Testing

1.1 Qu'est-ce qu'un A/B Test?

```
A/B Test = Expérience contrôlée randomisée
├── Groupe A (Contrôle): Version actuelle
├── Groupe B (Traitement): Nouvelle version
└── Comparaison: Mesurer la différence statistiquement significative
```

1.2 Quand utiliser l'A/B Testing en banque?

Cas d'usage	Exemple UniBank Haiti
Campagnes marketing	Email A vs Email B pour promotion crédit
Interface digitale	Nouveau design app mobile vs ancien
Processus crédit	Nouveau formulaire de demande de prêt
Tarification	Test de nouveaux frais de service
Communication	SMS de rappel vs appel téléphonique
Onboarding	Parcours d'ouverture de compte simplifié

1.3 Terminologie Essentielle

Terme	Définition	Exemple
Contrôle (A)	Groupe exposé à la version existante	Clients recevant l'email actuel
Traitement (B)	Groupe exposé à la nouvelle version	Clients recevant le nouvel email
Métrique primaire	KPI principal mesuré	Taux de conversion crédit
Métriques secondaires	KPIs complémentaires	Taux d'ouverture, temps de réponse
Effet minimal détectable (MDE)	Plus petite différence qu'on veut détecter	+2% de conversion
Puissance statistique	Probabilité de détecter un effet réel	80% standard
Niveau de signification (α)	Seuil de rejet de H_0	5% standard

---

Partie 2: Design Expérimental

2.1 Les 5 Étapes d'un A/B Test

```
1. HYPOTHÈSE
   └── "Le nouvel email augmentera le taux de souscription crédit de 5%"

2. DESIGN
   ├── Définir métrique primaire
   ├── Calculer taille d'échantillon
   └── Définir durée du test

3. RANDOMISATION
   └── Assigner aléatoirement les clients aux groupes

4. EXÉCUTION
   └── Collecter les données pendant la durée prévue

5. ANALYSE
   ├── Test statistique
   └── Décision: Déployer ou non
```

2.2 Calcul de la Taille d'Échantillon

Formule simplifiée pour proportions:

$$n = 2 \times [(Z\alpha/2 + Z\beta)^2 \times p(1-p)] / \delta^2$$

Où:
- n = taille par groupe
- Zα/2 = 1.96 pour α=5%
- Zβ = 0.84 pour puissance=80%
- p = proportion de base
- δ = effet minimal détectable (MDE)

En Python:

```python
from statsmodels.stats.power import tt_ind_solve_power
from statsmodels.stats.proportion import proportion_effectsize
import numpy as np

def sample_size_proportion(p_control, mde, alpha=0.05, power=0.80):
    """
    Calcule la taille d'échantillon nécessaire pour un A/B test.

    Args:
        p_control: Taux de conversion actuel (ex: 0.10 pour 10%)
        mde: Effet minimal détectable en points de pourcentage (ex: 0.02 pour +2%)
        alpha: Niveau de signification (default: 0.05)
        power: Puissance statistique (default: 0.80)

    Returns:
        Taille d'échantillon par groupe
    """
    p_treatment = p_control + mde
    effect_size = proportion_effectsize(p_treatment, p_control)
```

```
    n = tt_ind_solve_power(
        effect_size=effect_size,
        alpha=alpha,
        power=power,
        ratio=1.0,  # Groupes de même taille
        alternative='two-sided'
    )

    return int(np.ceil(n))

# Exemple UniBank: Email marketing crédit
# Taux actuel: 5%, on veut détecter +1.5%
n_per_group = sample_size_proportion(p_control=0.05, mde=0.015)
print(f"Taille nécessaire par groupe: {n_per_group}")
print(f"Total: {2 * n_per_group} clients")
```

2.3 Durée Minimale du Test

Règles importantes:

1. **Cycle complet:** Au minimum 1-2 semaines pour capturer les variations jour/semaine
2. **Effet de nouveauté:** Attendre que l'effet "nouveau" se stabilise
3. **Saisonnalité:** Éviter les périodes atypiques (fêtes, fin de mois pour salaires)

```
def duree_minimale_test(n_per_group, traffic_quotidien):
    """
    Estime la durée minimale du test.

    Args:
        n_per_group: Taille d'échantillon par groupe
        traffic_quotidien: Nombre de clients exposés par jour

    Returns:
        Durée en jours (minimum 7)
    """
    duree_base = np.ceil(2 * n_per_group / traffic_quotidien)
    return max(7, int(duree_base))  # Minimum 7 jours

# Exemple: 5000 clients par groupe, 500 clients/jour
duree = duree_minimale_test(5000, 500)
print(f"Durée minimale: {duree} jours")
```

Partie 3: Randomisation et Assignation

3.1 Méthodes de Randomisation

Méthode	Description	Quand utiliser
Simple	Tirage aléatoire pur	Grands échantillons homogènes
Stratifiée	Randomisation par segment	Groupes hétérogènes importants
Par cluster	Randomisation par agence/région	Éviter contamination

Méthode	Description	Quand utiliser
Séquentielle	Assignation au fil de l'eau	Tests continus

3.2 Implémentation Python

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

def randomize_ab_test(df, stratify_col=None, test_size=0.5, seed=42):
    """
    Randomise les clients pour un A/B test.

    Args:
        df: DataFrame avec les clients
        stratify_col: Colonne pour stratification (optionnel)
        test_size: Proportion groupe B (default: 0.5)
        seed: Graine aléatoire pour reproductibilité

    Returns:
        DataFrame avec colonne 'groupe' (A ou B)
    """
    if stratify_col:
        stratify = df[stratify_col]
    else:
        stratify = None

    group_a, group_b = train_test_split(
        df,
        test_size=test_size,
        random_state=seed,
        stratify=stratify
    )

    group_a = group_a.copy()
    group_b = group_b.copy()
    group_a['groupe'] = 'A'
    group_b['groupe'] = 'B'

    return pd.concat([group_a, group_b]).sort_index()

# Exemple: Stratification par segment client
df_clients['groupe'] = randomize_ab_test(
    df_clients,
    stratify_col='segment'
)['groupe']

# Vérification équilibre
print(df_clients.groupby(['groupe', 'segment']).size())
```

3.3 Vérification de la Randomisation (A/A Check)

Avant de lancer le test, vérifier que les groupes sont équilibrés:

```python
def verifier_equilibre(df, groupe_col='groupe', vars_to_check=None):
    """
    Vérifie l'équilibre entre groupes A et B.
    """
    if vars_to_check is None:
        vars_to_check = df.select_dtypes(include=[np.number]).columns

    results = []
    for var in vars_to_check:
        if var == groupe_col:
            continue

        group_a = df[df[groupe_col] == 'A'][var].dropna()
        group_b = df[df[groupe_col] == 'B'][var].dropna()

        # t-test pour vérifier l'équilibre
        from scipy import stats
        stat, pval = stats.ttest_ind(group_a, group_b)

        results.append({
            'variable': var,
            'mean_A': group_a.mean(),
            'mean_B': group_b.mean(),
            'diff_%': (group_b.mean() - group_a.mean()) / group_a.mean() * 100,
            'p_value': pval,
            'equilibre': 'OK' if pval > 0.05 else 'ATTENTION'
        })

    return pd.DataFrame(results)

# Vérifier avant de lancer
equilibre = verifier_equilibre(df_clients, vars_to_check=['age', 'anciennete', 'solde_moye
print(equilibre)
```

Partie 4: Analyse des Résultats

4.1 Test Statistique pour Proportions (Chi-carré / Z-test)

```python
from scipy import stats
from statsmodels.stats.proportion import proportions_ztest

def analyser_ab_test_proportion(conversions_a, n_a, conversions_b, n_b, alpha=0.05):
    """
    Analyse les résultats d'un A/B test pour des proportions.

    Args:
        conversions_a: Nombre de conversions groupe A
        n_a: Taille groupe A
        conversions_b: Nombre de conversions groupe B
        n_b: Taille groupe B
        alpha: Seuil de signification
```

```
    Returns:
        Dictionnaire avec résultats
    """
    # Taux de conversion
    p_a = conversions_a / n_a
    p_b = conversions_b / n_b

    # Différence relative
    lift = (p_b - p_a) / p_a * 100

    # Z-test pour proportions
    count = np.array([conversions_b, conversions_a])
    nobs = np.array([n_b, n_a])
    z_stat, p_value = proportions_ztest(count, nobs)

    # Intervalle de confiance sur la différence
    se = np.sqrt(p_a*(1-p_a)/n_a + p_b*(1-p_b)/n_b)
    ci_lower = (p_b - p_a) - 1.96 * se
    ci_upper = (p_b - p_a) + 1.96 * se

    # Décision
    significatif = p_value < alpha

    return {
        'taux_controle': f"{p_a:.2%}",
        'taux_traitement': f"{p_b:.2%}",
        'lift': f"{lift:+.1f}%",
        'p_value': f"{p_value:.4f}",
        'ic_95': f"[{ci_lower:.4f}, {ci_upper:.4f}]",
        'significatif': significatif,
        'decision': 'DÉPLOYER' if significatif and lift > 0 else 'NE PAS DÉPLOYER'
    }

# Exemple: Campagne email crédit UniBank
resultats = analyser_ab_test_proportion(
    conversions_a=250,  # 5% de 5000
    n_a=5000,
    conversions_b=325,  # 6.5% de 5000
    n_b=5000
)

for key, value in resultats.items():
    print(f"{key}: {value}")
```

4.2 Test pour Moyennes (t-test)

```
def analyser_ab_test_moyenne(values_a, values_b, alpha=0.05):
    """
    Analyse les résultats d'un A/B test pour des moyennes.

    Args:
        values_a: Valeurs groupe A
        values_b: Valeurs groupe B
        alpha: Seuil de signification
```

```
    Returns:
        Dictionnaire avec résultats
    """
    mean_a = np.mean(values_a)
    mean_b = np.mean(values_b)

    # t-test
    t_stat, p_value = stats.ttest_ind(values_a, values_b)

    # Lift
    lift = (mean_b - mean_a) / mean_a * 100

    # Intervalle de confiance (bootstrap)
    n_bootstrap = 10000
    diffs = []
    for _ in range(n_bootstrap):
        sample_a = np.random.choice(values_a, size=len(values_a), replace=True)
        sample_b = np.random.choice(values_b, size=len(values_b), replace=True)
        diffs.append(np.mean(sample_b) - np.mean(sample_a))

    ci_lower = np.percentile(diffs, 2.5)
    ci_upper = np.percentile(diffs, 97.5)

    significatif = p_value < alpha

    return {
        'moyenne_controle': f"{mean_a:.2f}",
        'moyenne_traitement': f"{mean_b:.2f}",
        'lift': f"{lift:+.1f}%",
        'p_value': f"{p_value:.4f}",
        'ic_95_diff': f"[{ci_lower:.2f}, {ci_upper:.2f}]",
        'significatif': significatif,
        'decision': 'DÉPLOYER' if significatif and lift > 0 else 'NE PAS DÉPLOYER'
    }

# Exemple: Montant moyen de dépôt après nouvelle interface
groupe_a = np.random.normal(50000, 15000, 3000)  # HTG
groupe_b = np.random.normal(52500, 15000, 3000)  # +5%
resultats = analyser_ab_test_moyenne(groupe_a, groupe_b)
```

4.3 Interprétation des Résultats

Matrice de décision:

p-value	Lift	Décision
< 0.05	> 0	Déployer B
< 0.05	< 0	Garder A
≥ 0.05	-	Pas de conclusion, prolonger ou abandonner

Attention aux pièges:

1. **Peeking:** Ne pas regarder les résultats avant la fin prévue

2. **Multiple testing:** Corriger pour tests multiples (Bonferroni)
3. **Effet de nouveauté:** L'effet peut diminuer avec le temps
4. **Segments cachés:** Analyser par segment peut révéler des effets différents

---

Partie 5: Tests Multi-Variés et Avancés

5.1 Tests A/B/n (Plus de 2 variantes)

```python
from scipy.stats import chi2_contingency

def analyser_ab_n_test(data):
    """
    Analyse un test avec plusieurs variantes.

    Args:
        data: Dict {variante: (conversions, total)}

    Returns:
        Résultats avec meilleure variante
    """
    # Créer table de contingence
    variantes = list(data.keys())
    conversions = [data[v][0] for v in variantes]
    non_conversions = [data[v][1] - data[v][0] for v in variantes]

    table = np.array([conversions, non_conversions])

    # Test Chi-carré
    chi2, p_value, dof, expected = chi2_contingency(table)

    # Taux par variante
    taux = {v: data[v][0] / data[v][1] for v in variantes}
    meilleure = max(taux, key=taux.get)

    return {
        'chi2': chi2,
        'p_value': p_value,
        'taux_par_variante': taux,
        'meilleure_variante': meilleure,
        'significatif': p_value < 0.05
    }

# Exemple: Test de 3 emails différents
resultats = analyser_ab_n_test({
    'Email_Standard': (250, 5000),
    'Email_Personnalise': (310, 5000),
    'Email_Urgence': (285, 5000)
})
```

5.2 Tests Séquentiels (Bayésien)

```python
def ab_test_bayesien(conversions_a, n_a, conversions_b, n_b, n_samples=100000):
    """
    A/B test avec approche bayésienne.

    Avantages:
    - Peut s'arrêter tôt si résultat clair
    - Fournit probabilité que B > A
    """
    from scipy.stats import beta

    # Priors non-informatifs
    alpha_prior = 1
    beta_prior = 1

    # Postérieurs
    alpha_a = alpha_prior + conversions_a
    beta_a = beta_prior + n_a - conversions_a
    alpha_b = alpha_prior + conversions_b
    beta_b = beta_prior + n_b - conversions_b

    # Échantillonnage
    samples_a = beta.rvs(alpha_a, beta_a, size=n_samples)
    samples_b = beta.rvs(alpha_b, beta_b, size=n_samples)

    # Probabilité que B > A
    prob_b_superior = (samples_b > samples_a).mean()

    # Lift espéré
    expected_lift = (samples_b.mean() - samples_a.mean()) / samples_a.mean() * 100

    return {
        'prob_B_superieur': f"{prob_b_superior:.1%}",
        'lift_espere': f"{expected_lift:+.1f}%",
        'decision': 'DÉPLOYER B' if prob_b_superior > 0.95 else
                    'GARDER A' if prob_b_superior < 0.05 else 'CONTINUER TEST'
    }
```

Partie 6: Cas Pratiques Bancaires UniBank Haiti

6.1 Cas 1: Optimisation Email Marketing Crédit

Contexte: UniBank veut améliorer le taux de souscription aux prêts personnels.

```python
# Situation
taux_actuel = 0.05  # 5% de conversion
mde = 0.01  # Détecter +1 point de pourcentage
traffic_jour = 1000  # Clients recevant l'email par jour

# Calcul taille échantillon
n = sample_size_proportion(taux_actuel, mde)
print(f"Taille par groupe: {n}")
```

```python
# Durée estimée
duree = duree_minimale_test(n, traffic_jour // 2)  # /2 car split 50/50
print(f"Durée: {duree} jours")

# Après le test
resultats = analyser_ab_test_proportion(
    conversions_a=180, n_a=3600,  # Contrôle
    conversions_b=234, n_b=3600   # Nouvel email
)
print(resultats)
```

6.2 Cas 2: Nouvelle Interface App Mobile

Contexte: Test du nouveau parcours de transfert d'argent.

```python
# Métriques multiples
metriques = {
    'taux_completion': {'a': (850, 1000), 'b': (910, 1000)},
    'temps_moyen_sec': {'a': 45.2, 'b': 38.7},
    'erreurs': {'a': (75, 1000), 'b': (52, 1000)}
}

# Analyse taux de complétion
result_completion = analyser_ab_test_proportion(850, 1000, 910, 1000)

# Analyse erreurs (on veut MOINS d'erreurs)
result_erreurs = analyser_ab_test_proportion(75, 1000, 52, 1000)

# Décision globale
print("Complétion:", result_completion['decision'])
print("Erreurs:", 'DÉPLOYER' if result_erreurs['significatif'] and
      result_erreurs['lift'].startswith('-') else 'Pas d\'amélioration')
```

6.3 Cas 3: Test Tarification Frais de Service

Contexte: Impact de nouveaux frais sur le churn.

```python
# ATTENTION: Tests de prix nécessitent précautions éthiques
# - Durée limitée
# - Communication transparente
# - Pas de discrimination

# Analyse churn après 3 mois
churn_control = 0.05  # 5% churn avec anciens frais
churn_test = 0.07     # 7% churn avec nouveaux frais

# Même si significatif, considérer l'impact business
revenu_additionnel = 5000000  # HTG par mois
clients_perdus = 0.02 * 100000  # 2% de plus × base clients
valeur_client = 50000  # LTV moyen

perte_churn = clients_perdus * valeur_client
print(f"Gain frais: {revenu_additionnel:,.0f} HTG")
```

```
print(f"Perte churn: {perte_churn:,.0f} HTG")
print(f"Impact net: {revenu_additionnel - perte_churn:,.0f} HTG")
```

Partie 7: Pièges et Bonnes Pratiques

7.1 Erreurs Courantes

Erreur	Conséquence	Solution
Peeking	Faux positifs	Définir durée à l'avance
Arrêt précoce	Conclusions fausses	Respecter taille échantillon
Multiple testing	Inflation erreur Type I	Correction Bonferroni
Selection bias	Groupes non comparables	Randomisation rigoureuse
Effet réseau	Contamination	Randomisation par cluster
Effet saisonnier	Résultats biaisés	Test sur période complète

7.2 Checklist Avant Lancement

```
## Checklist A/B Test UniBank

### Préparation
- [ ] Hypothèse clairement formulée
- [ ] Métrique primaire définie
- [ ] Métriques secondaires listées
- [ ] Taille d'échantillon calculée
- [ ] Durée minimale déterminée
- [ ] Randomisation vérifiée (A/A check)

### Éthique
- [ ] Pas de discrimination (âge, genre, localisation)
- [ ] Impact client acceptable
- [ ] Conformité réglementaire vérifiée

### Exécution
- [ ] Tracking correctement implémenté
- [ ] Alertes configurées pour anomalies
- [ ] Documentation de tout changement

### Analyse
- [ ] Attente de la durée complète
- [ ] Test statistique approprié
- [ ] Analyse par segment
- [ ] Documentation des résultats
```

7.3 Correction pour Tests Multiples (Bonferroni)

```
def correction_bonferroni(p_values, alpha=0.05):
    """
    Corrige les p-values pour tests multiples.
    """
    n_tests = len(p_values)
    alpha_corrige = alpha / n_tests
```

```
    results = []
    for metric, pval in p_values.items():
        results.append({
            'metric': metric,
            'p_value': pval,
            'seuil_corrige': alpha_corrige,
            'significatif': pval < alpha_corrige
        })

    return pd.DataFrame(results)

# Exemple avec 3 métriques
p_values = {
    'taux_conversion': 0.023,
    'panier_moyen': 0.048,
    'temps_session': 0.012
}

resultats_corriges = correction_bonferroni(p_values)
print(resultats_corriges)
# Avec Bonferroni (α=0.05/3=0.017), seul temps_session reste significatif
```

Résumé et Mnémotechniques

Formules Clés

```
Taille échantillon ≈ 16 × σ² / δ²  (règle rapide)
Puissance = 1 - β (typiquement 80%)
MDE = Effet minimal à détecter
Durée ≥ 7 jours (minimum absolu)
```

Mnémotechniques

Concept	Mnémotechnique
Étapes A/B Test	**HDREA** - Hypothèse, Design, Randomisation, Exécution, Analyse
Erreurs Type I/II	**FP-FN** - Faux Positif (α) / Faux Négatif (β)
Puissance	**80-20** - 80% puissance, 20% risque de manquer l'effet
Signification	**5%** - Seuil standard pour rejeter H_0

Points Clés Banking

1. **Toujours stratifier** par segment client (Retail/Premium/Corporate)
2. **Attention aux effets réseau** (ex: transferts entre clients)
3. **Mesurer l'impact business** en plus de la significativité statistique
4. **Documenter pour la conformité** réglementaire

Objectif: Prendre des décisions data-driven plutôt que basées sur l'intuition ou la hiérarchie.