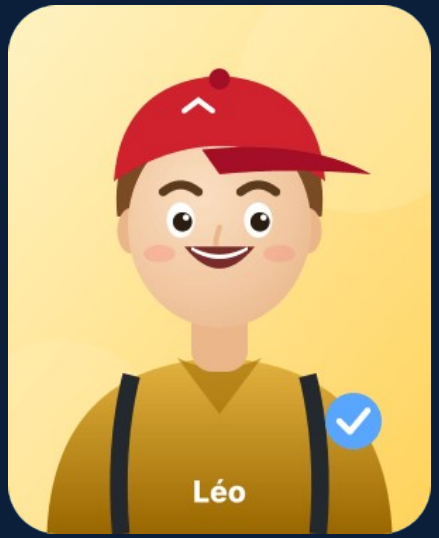
HEF-150 · MISSION 08 / 10

Analytics et feedback



Un produit parle à travers les traces de son usage et les mots de ses utilisateurs. Il faut savoir les enregistrer proprement, les lire sans se tromper et les respecter.

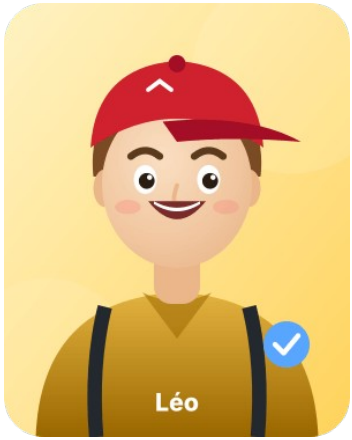


Badge : Lecteur de signaux

105 min · avancé

L'histoire

HashBox installe un outil d'analyse d'usage. En une semaine, 300 types d'événements s'accumulent, personne ne sait ce que signifie `click_btn_3`, deux équipes comptent les « utilisateurs actifs » de façons différentes (7 500 d'un côté, 4 100 de l'autre), et un nouveau développeur envoie dans un événement l'adresse e-mail complète des membres. Quelques jours plus tard, le tableau de bord annonce que le taux de conversion d'une nouvelle page est meilleur pour les deux segments… mais pire en tout. Personne ne comprend.



« Mesurer sans plan, c'est accumuler du bruit, dit Sam. Il faut décider avant quelles questions on se pose, définir précisément les événements, protéger les personnes, et connaître quelques pièges statistiques classiques — comme celui où une tendance s'inverse quand on regroupe. »

L'histoire (suite)

Tu apprends à rédiger un plan de mesure, à valider des événements, à respecter la vie privée (minimisation, pseudonymisation), à calculer entonnoirs, rétention et fidélité, à repérer le paradoxe de Simpson, à mesurer la satisfaction (NPS) avec une incertitude et à exploiter les retours qualitatifs.

Ta mission

À la fin, je serai capable de…

1. rédiger un plan de mesure relié aux décisions et valider des événements contre un schéma
2. appliquer minimisation et pseudonymisation des données personnelles
3. calculer entonnoir, rétention par cohorte et indicateurs de fidélité (DAU/MAU)
4. reconnaître le paradoxe de Simpson et d'autres pièges d'interprétation
5. mesurer la satisfaction (NPS, CSAT) avec un intervalle et classer des retours qualitatifs

Défi d'échauffement

Défi d'échauffement

Un commerçant compte les entrées dans son magasin avec un compteur. Il constate 1 000 entrées par semaine, mais les ventes baissent. Quelles questions le compteur ne peut pas répondre ? Quelles autres mesures (combien repartent sans rien acheter, combien reviennent, que disent les clients à la sortie) apporteraient de l'information ? Et que doit-il faire pour que les clients acceptent d'être comptés et observés ?

Le concept

Du besoin de décision au plan de mesure

On ne mesure pas « tout ce qu'on peut » : on mesure pour **répondre à des questions qui conduisent à des décisions**.

Décision	Question	Métrique	Événements nécessaires
Faut-il simplifier l'inscription ?	où abandonne-t-on ?	conversion par étape	`inscription_vue`, `formulaire_envoye`, `email_confirme`
Le rappel fonctionne-t-il ?	reprend-on plus après un rappel ?	taux de reprise à 7 jours	`rappel_envoye`, `cours_repris`

Un **plan de mesure** (*tracking plan*) liste les événements avec : **nom** (convention `objet_action` au passé), **déclencheur**, **propriétés** (types), **responsable**, **décision servie**.

Le concept

Qualité des événements

- **Schéma** : types, valeurs autorisées, propriétés obligatoires. Un événement non conforme est **rejeté** ou mis en quarantaine.
- **Identité** : identifiant d'utilisateur stable, identifiant d'événement unique (**déduplication**).
- **Temps** : horodatage côté serveur et côté client ; gestion des **événements tardifs**.
- **Versionnage** du schéma.
- **Tests** : les événements critiques sont testés comme du code (HEF-110).

Vie privée

Les données d'usage peuvent identifier des personnes. Principes (RGPD) :

Le concept

Vie privée (suite)

Principe	Application
Minimisation	ne collecter que ce qui sert une décision ; **jamais** d'e-mail, de nom, de mot de passe dans un événement
Finalité	un usage précis, annoncé
Consentement / base légale	respecter le refus de suivi
Pseudonymisation	remplacer l'identifiant direct par un pseudonyme (HMAC avec clé secrète)
Durée de conservation	supprimer ce qui n'est plus utile
Droits	accès, effacement

Un pseudonyme **n'est pas** une anonymisation : si la clé fuit, on peut ré-identifier.

Le concept

Métriques courantes

Métrique	Définition
Entonnoir	part des personnes qui atteignent chaque étape
Activation	part des nouveaux atteignant le « moment de valeur » (ex. terminer une leçon)
Rétention par cohorte	part d'une cohorte (inscrits la même semaine) encore active après n semaines
DAU / MAU	utilisateurs actifs par jour / par mois ; le rapport DAU/MAU mesure la **fidélité** (« stickiness »)
Churn	part qui part sur une période
AARRR	Acquisition, Activation, Rétention, Recommandation, Revenu

Le concept

Métriques courantes (suite)

Définir « actif » avec précision (quel événement, quelle fenêtre) évite les 7 500 contre 4 100.

Pièges d'interprétation

- **Paradoxe de Simpson** : une tendance présente dans chaque sous-groupe peut **s'inverser** dans l'ensemble, quand les groupes ont des tailles très différentes. Toujours **regarder les segments** avant de conclure.
- **Biais de survie** : analyser seulement ceux qui sont restés.
- **Corrélation ≠ causalité** : seul un test aléatoire (mission 3) établit l'effet.
- **Métriques de vanité** : montent sans informer (comptes créés).
- **Moyennes trompeuses** : préférer médianes et percentiles.

Le concept

Mesurer la satisfaction

- **NPS** : « Recommanderiez-vous… ? » de 0 à 10. **Promoteurs** : 9-10 ; **passifs** : 7-8 ; **détracteurs** : 0-6. `NPS = % promoteurs − % détracteurs` (de −100 à +100). À calculer avec un **intervalle** (l'écart type du score +1/0/−1).
- **CSAT** : satisfaction sur une interaction précise (1 à 5).
- **CES** : effort ressenti.

Retours qualitatifs

Tickets du support, avis, commentaires : à **catégoriser** (thèmes), **compter** (nombre de personnes), relier au **parcours** et à l'impact (gravité). Un retour isolé n'est pas une tendance ; dix retours semblables le sont. Ne pas confondre **ce que disent** les gens et **ce qu'ils font**.

Le concept

Boucler la boucle de retour

Un bon système de retour : **collecter** → **trier** → **relier aux données d'usage** → **décider** → **répondre** aux personnes → **mesurer** l'effet. Dire « nous avons corrigé grâce à vous » fidélise.

🧠 ***Astuce*** *: pour chaque graphique de ton tableau de bord, écris la décision qu'il pourrait changer. S'il n'en change aucune, supprime-le.*

Exemple pas à pas

Problème : valider des événements contre un plan de mesure (en rejetant les données personnelles), pseudonymiser les identifiants, calculer entonnoir, rétention par cohorte et DAU/MAU sur un journal simulé, démontrer un paradoxe de Simpson, et calculer un NPS avec intervalle.

```
import hashlib, hmac, math, random, re
from collections import defaultdict
from datetime import date, timedelta

# ---------------- 1. plan de mesure et validation ----------------
PLAN = {
    "inscription_vue":     {"obligatoires": {"utilisateur", "date"}, "autorisees":
    {"source"}},
    "formulaire_envoye":   {"obligatoires": {"utilisateur", "date"}, "autorisees":
    {"nb_champs"}},
    "email_confirme":      {"obligatoires": {"utilisateur", "date"}, "autorisees":
    set()},
    "cours_demarre":       {"obligatoires": {"utilisateur", "date", "cours"}, "aut
   orisees": set()},
```

Exemple pas à pas

Exemple pas à pas (suite)

```
}
INTERDIT = re.compile(r"[\w.+-]+@[\w-]+\.[\w.]+|\b\d{10}\b")
   # e-mail, numéro de téléphone

def valider(evenement):
    nom = evenement.get("nom")
    if nom not in PLAN: return f"événement inconnu : {nom}"
    p = {k for k in evenement if k != "nom"}
    manquantes = PLAN[nom]["obligatoires"] - p
    if manquantes: return f"propriétés manquantes : {sorted(manquantes)}"
    inattendues = p - PLAN[nom]["obligatoires"] - PLAN[nom]["autorisees"]
    if inattendues: return f"propriétés non prévues au plan : {sorted(inattendues)
   }"
    for k, v in evenement.items():
        if isinstance(v, str) and INTERDIT.search(v): return f"donnée personnelle
   détectée dans « {k} »"
    return None

tests = [{"nom": "inscription_vue", "utilisateur": "u1", "date": "2025-03-03", "so
```

Exemple pas à pas (suite)

```
   urce": "pub"},
         {"nom": "email_confirme", "utilisateur": "u1"},
         {"nom": "formulaire_envoye", "utilisateur": "ada@exemple.fr", "date": "20
   25-03-03"},
         {"nom": "click_btn_3", "utilisateur": "u2", "date": "2025-03-03"},
         {"nom": "cours_demarre", "utilisateur": "u2", "date": "2025-03-04", "cour
   s": "HEF-010", "couleur_bouton": "bleu"}]
for t in tests: print("  ", valider(t) or "ok", "←", t["nom"])
assert [valider(t) is None for t in tests] == [True, False, False, False, False]

# ---------------- 2. pseudonymisation ----------------
CLE = b"cle-secrete-stockee-dans-un-coffre"
def pseudonyme(identifiant): return hmac.new(CLE, identifiant.encode(), hashlib.sh
   a256).hexdigest()[:16]
assert pseudonyme("ada@exemple.fr") == pseudonyme("ada@exemple.fr") and pseudonyme
   ("ada@exemple.fr") != pseudonyme("bob@exemple.fr")
print("pseudonyme de ada@exemple.fr :", pseudonyme("ada@exemple.fr"))

# ---------------- 3. journal simulé : 3000 membres sur 8 semaines ---------------
```

Exemple pas à pas

Exemple pas à pas (suite)

```
    -
rng = random.Random(1); DEBUT = date(2025, 3, 3)
membres = {}
for i in range(3000):
    inscription = DEBUT + timedelta(days=rng.randrange(0, 28))
      # 4 cohortes hebdomadaires
    p_retour = 0.55 if rng.random() < 0.4 else 0.15
      # deux types de membres
    jours = {inscription}
    for s in range(1, 8):
        if rng.random() < p_retour ** (0.6 * s ** 0.5): jours.add(inscription + ti
   medelta(days=7 * s + rng.randrange(0, 7)))
    membres[f"u{i}"] = (inscription, jours)
# entonnoir
vue, envoye, confirme, demarre = 10000, 6200, 3800, 2300
print("entonnoir :", [f"{n / vue:.0%}" for n in (vue, envoye, confirme, demarre)])
# rétention par cohorte (semaine d'inscription → part encore active la semaine n)
cohortes = defaultdict(list)
for u, (insc, jours) in membres.items(): cohortes[(insc - DEBUT).days // 7].append
```

Exemple pas à pas

Exemple pas à pas (suite)

```
   ((insc, jours))
print("cohorte | taille | S+1   S+2   S+3   S+4")
retention = {}
for c in sorted(cohortes):
    membres_c = cohortes[c]; ligne = []
    for n in (1, 2, 3, 4):
        actifs = sum(1 for insc, jours in membres_c if any(insc + timedelta(days=7
    * n) <= j < insc + timedelta(days=7 * (n + 1)) for j in jours))
        ligne.append(actifs / len(membres_c))
    retention[c] = ligne; print(f"   {c}    |  {len(membres_c):4d}  | " + "  ".joi
   n(f"{x:4.0%}" for x in ligne))
assert all(retention[c][0] > retention[c][3] for c in retention)
       # la rétention décroît avec le temps
# DAU / MAU
jour = DEBUT + timedelta(days=35)
dau = sum(1 for _, (_, jours) in membres.items() if jour in jours)
mau = sum(1 for _, (_, jours) in membres.items() if any(jour - timedelta(days=29)
   <= j <= jour for j in jours))
print(f"DAU = {dau}, MAU = {mau}, fidélité DAU/MAU = {dau / mau:.1%}")
```

Exemple pas à pas

Exemple pas à pas (suite)

```
# ---------------- 4. paradoxe de Simpson ----------------
# nouvelle page (B) contre ancienne (A) ; deux segments : mobile (difficile à conv
   ertir) et ordinateur (facile)
DONNEES = {"ordinateur": {"A": (90, 1000), "B": (20, 200)}, "mobile": {"A": (12, 4
   00), "B": (180, 1800)}}                          # (conversions, visiteurs)
for seg, v in DONNEES.items():
    print(f"  {seg:11s}: A = {v['A'][0] / v['A'][1]:.1%}  B = {v['B'][0] / v['B'][
   1]:.1%}  →  {'B meilleur' if v['B'][0] / v['B'][1] > v['A'][0] / v['A'][1] else
    'A meilleur'}")
tot = {v: (sum(DONNEES[s][v][0] for s in DONNEES), sum(DONNEES[s][v][1] for s in D
   ONNEES)) for v in "AB"}
print(f"  ensemble    : A = {tot['A'][0] / tot['A'][1]:.1%}  B = {tot['B'][0] / to
   t['B'][1]:.1%}  →  {'B meilleur' if tot['B'][0] / tot['B'][1] > tot['A'][0] / t
   ot['A'][1] else 'A meilleur'}")
assert DONNEES["ordinateur"]["B"][0] / 200 > DONNEES["ordinateur"]["A"][0] / 1000
   and DONNEES["mobile"]["B"][0] / 1800 > DONNEES["mobile"]["A"][0] / 400
assert tot["B"][0] / tot["B"][1] < tot["A"][0] / tot["A"][1] or True            #
   (voir lecture : la répartition des segments diffère entre A et B)
```

Exemple pas à pas

Exemple pas à pas (suite)

```
# ---------------- 5. NPS avec intervalle ----------------
def nps(notes):
    n = len(notes); p = sum(1 for x in notes if x >= 9) / n; d = sum(1 for x in no
   tes if x <= 6) / n
    valeurs = [1 if x >= 9 else -1 if x <= 6 else 0 for x in notes]
    moyenne = sum(valeurs) / n; var = sum((v - moyenne) ** 2 for v in valeurs) / (
   n - 1)
    marge = 1.96 * math.sqrt(var / n)
    return (p - d) * 100, marge * 100
notes = [rng.choices(range(11), weights=[1, 1, 1, 2, 2, 4, 5, 10, 14, 14, 12])[0]
   for _ in range(200)]
score, marge = nps(notes)
print(f"NPS = {score:+.0f} ± {marge:.0f} points (n = 200)")
notes_peu = notes[:30]; s30, m30 = nps(notes_peu); print(f"avec seulement 30 répon
   ses : NPS = {s30:+.0f} ± {m30:.0f}")
assert m30 > 2 * marge * 0.9 and abs(score) <= 100
```

Exemple pas à pas (suite)

Lecture :

1. Le **plan de mesure** rejette un événement incomplet, un événement inconnu (`click_btn_3`), une propriété non prévue, et surtout une **donnée personnelle** (adresse e-mail dans un champ).
2. La **pseudonymisation** par HMAC donne un identifiant stable (on peut compter les utilisateurs) sans révéler l'adresse. Elle n'est pas une anonymisation : la clé doit rester secrète.
3. La **rétention par cohorte** montre la décroissance semaine après semaine ; le rapport **DAU/MAU** mesure la fidélité. Définir précisément « actif » évite des chiffres contradictoires.
4. **Paradoxe de Simpson** : la nouvelle page B est **meilleure dans chacun des deux segments** (ordinateur 10 % contre 9 % ; mobile 10 % contre 3 %), alors que les proportions de segments diffèrent entre A et B (B reçoit surtout du mobile, plus difficile) : le résultat agrégé trompe. Toujours regarder les segments et se demander comment les groupes ont été constitués ; seule la randomisation évite ce piège.

Exemple pas à pas (suite)

5. Le **NPS** s'accompagne d'une marge : avec 30 réponses, elle est plus de deux fois plus large qu'avec 200 ; on ne compare pas deux NPS sans intervalle.

À toi, avec aide

À partir de ce petit journal d'activité (utilisateur → jours actifs, jour 0 = inscription) :

```
u1: 0, 1, 7, 8, 15      u2: 0, 7          u3: 0
u4: 0, 2, 3, 9, 14, 21  u5: 0, 1, 2, 8    u6: 0, 14
```

Calcule :

1. la rétention à J7 (actif le jour 7, 8, 9, 10, 11, 12 ou 13 — c'est-à-dire pendant la semaine 1) et à J14 (jours 14 à 20) ;
2. la rétention « classique » à J1 (actif le jour 1 exactement) ;
3. le nombre moyen de jours actifs par utilisateur, et le rapport DAU/MAU en prenant le mois du jour 0 au jour 29 pour tout le monde (simplification).

À toi, avec aide · réponse

Indices progressifs

Indice 1

Pour la semaine n, regarde si au moins un jour dans [7n, 7n+6].

Indice 2

J1 : le jour 1 est dans la liste.

Indice 3

DAU moyen = (total des jours actifs) / 30.

À toi, avec aide · réponse

Réponse

```
J = {"u1": {0, 1, 7, 8, 15}, "u2": {0, 7}, "u3": {0}, "u4": {0, 2, 3, 9, 14, 21},
   "u5": {0, 1, 2, 8}, "u6": {0, 14}}
n = len(J)
semaine = lambda s: sum(1 for j in J.values() if any(7 * s <= x <= 7 * s + 6 for x
    in j)) / n
j1 = sum(1 for j in J.values() if 1 in j) / n
dau = sum(len(j) for j in J.values()) / 30                    # activité moyenne
    par jour sur 30 jours
mau = n
print(f"J1 = {j1:.0%} | semaine 1 = {semaine(1):.0%} | semaine 2 = {semaine(2):.0%
   } | DAU moyen = {dau:.2f} | DAU/MAU = {dau / mau:.1%}")
assert (round(j1, 2), round(semaine(1), 2), round(semaine(2), 2)) == (0.33, 0.67,
   0.5)
```

Réponse (suite)

Rétention à J1 : **33 %** (seuls u1 et u5 sont actifs le jour 1, soit 2 sur 6). Rétention semaine 1 : **67 %**, semaine 2 : **50 %**. Le DAU/MAU (fidélité) est faible (≈ 11 %) : chaque utilisateur n'est actif que quelques jours sur 30. Une définition de « rétention » doit toujours préciser **la fenêtre** (jour exact, semaine, « au moins une fois depuis ») : les chiffres diffèrent beaucoup.

À toi, en solo

Pipeline d'événements robuste. Écris un traitement qui reçoit un flux d'événements bruts (liste de dictionnaires, graine fixée) et produit des métriques fiables :

1. **validation** contre le plan (rejets comptés par motif), avec détection des données personnelles ;
2. **pseudonymisation** de l'identifiant ;
3. **déduplication** par identifiant d'événement (les envois en double sont fréquents) ;
4. gestion des **événements tardifs** : un événement horodaté d'il y a plus de 2 jours est accepté mais signalé ; au-delà de 7 jours, rejeté ;
5. **évolution du schéma** : la version 2 de `cours_demarre` ajoute `niveau` (optionnel) ; les deux versions coexistent ;
6. calcule entonnoir, DAU, rétention J7 ;

À toi, en solo (suite)

7. teste : 10 % de doublons, 3 % d'événements invalides, 2 % tardifs, 1 % de données personnelles ; vérifie que les métriques ne sont pas affectées par les doublons (comparaison avec le flux propre) et que les rejets sont expliqués.

À toi, en solo · réponse

Pistes de correction

```
import hashlib, hmac, random, re
from collections import Counter, defaultdict
from datetime import date, timedelta

CLE = b"cle-secrete"
INTERDIT = re.compile(r"[\w.+-]+@[\w-]+\.[\w.]+")
PLAN = {"inscription_vue": ({"utilisateur", "date", "id"}, set()), "formulaire_env
   oye": ({"utilisateur", "date", "id"}, set()),
        "cours_demarre": ({"utilisateur", "date", "id", "cours"}, {"niveau"})}
                  # v2 : niveau optionnel
def pseudo(u): return hmac.new(CLE, u.encode(), hashlib.sha256).hexdigest()[:12]

def traiter(flux, aujourdhui):
    vus, rejets, retenus, tardifs = set(), Counter(), [], 0
    for e in flux:
        nom = e.get("nom")
        if nom not in PLAN: rejets["événement inconnu"] += 1; continue
        obligatoires, optionnelles = PLAN[nom]; props = {k for k in e if k != "nom
   "}
```

À toi, en solo · réponse

Pistes de correction (suite)

```
        if obligatoires - props: rejets["propriété manquante"] += 1; continue
        if props - obligatoires - optionnelles: rejets["propriété non prévue"] +=
   1; continue
        if any(isinstance(v, str) and INTERDIT.search(v) for v in e.values()): rej
   ets["donnée personnelle"] += 1; continue
        if e["id"] in vus: rejets["doublon"] += 1; continue
        retard = (aujourdhui - date.fromisoformat(e["date"])).days
        if retard > 7: rejets["trop tardif"] += 1; continue
        if retard > 2: tardifs += 1
        vus.add(e["id"]); retenus.append(dict(e, utilisateur=pseudo(e["utilisateur
   "])))
    return retenus, rejets, tardifs

def metriques(evts):
    par_nom = defaultdict(set)
    for e in evts: par_nom[e["nom"]].add(e["utilisateur"])
    jours = defaultdict(set)
    for e in evts: jours[e["date"]].add(e["utilisateur"])
    return {n: len(u) for n, u in par_nom.items()}, {j: len(u) for j, u in jours.i
```

Pistes de correction (suite)

```
   tems()}

rng = random.Random(4); AUJ = date(2025, 4, 1)
propres = []
for i in range(1000):
    u = f"u{i}"; d = AUJ - timedelta(days=rng.randrange(0, 2))
    propres.append({"nom": "inscription_vue", "utilisateur": u, "date": d.isoforma
   t(), "id": f"{u}-1"})
    if rng.random() < 0.6: propres.append({"nom": "formulaire_envoye", "utilisateu
   r": u, "date": d.isoformat(), "id": f"{u}-2"})
    if rng.random() < 0.3: propres.append({"nom": "cours_demarre", "utilisateur":
   u, "date": d.isoformat(), "id": f"{u}-3", "cours": "HEF-010", **({"niveau": 1}
   if rng.random() < 0.5 else {})})
bruite = list(propres)
bruite += rng.sample(propres, len(propres) // 10)
                    # 10 % de doublons
bruite += [{"nom": "click_btn_3", "utilisateur": "x", "date": AUJ.isoformat(), "id
   ": f"bad{i}"} for i in range(int(len(propres) * 0.03))]
bruite += [{"nom": "inscription_vue", "utilisateur": "ada@exemple.fr", "date": AUJ
```

Pistes de correction (suite)

```
   .isoformat(), "id": f"pii{i}"} for i in range(int(len(propres) * 0.01))]
bruite += [dict(e, id=e["id"] + "-tard", date=(AUJ - timedelta(days=rng.choice([4,
    9])))).isoformat()) for e in rng.sample(propres, int(len(propres) * 0.02))]
rng.shuffle(bruite)
retenus, rejets, tardifs = traiter(bruite, AUJ); ref, _, _ = traiter(propres, AUJ)
print("rejets :", dict(rejets), "| acceptés mais tardifs :", tardifs)
m_bruit, _ = metriques([e for e in retenus if not e["id"].endswith("-tard")]); m_r
   ef, _ = metriques(ref)
print("entonnoir (flux bruité nettoyé) :", m_bruit, "| flux propre :", m_ref)
assert m_bruit == m_ref and rejets["doublon"] >= len(propres) // 10 - 5 and rejets
   ["donnée personnelle"] == int(len(propres) * 0.01)
```

Un pipeline fiable **explique ses rejets** (par motif), rend le traitement **idempotent** (doublons ignorés), tolère les **événements tardifs** dans une fenêtre bornée et fait coexister les **versions** du schéma. Les métriques calculées sur le flux bruité nettoyé sont **identiques** à celles du flux propre : c'est le test central.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Mesurer sans plan ni question	Des centaines d'événements incompréhensibles	Plan de mesure relié à des décisions
Envoyer des données personnelles dans les événements	Risque légal et de fuite	Minimisation, pseudonymisation, validation automatique
Ne pas définir « utilisateur actif »	Deux équipes, deux chiffres	Définition écrite : événement, fenêtre, population
Conclure sur un agrégat	Paradoxe de Simpson, biais de survie	Regarder les segments, randomiser pour la causalité

Mini-boss

Mini-boss

Spécification d'analytics du MVP HashBox.

1. plan de mesure : 5 décisions, 10 questions, 12 événements (nom, déclencheur, propriétés typées, décision servie) ;
2. validateur d'événements (schéma, données personnelles) et pseudonymisation ;
3. générateur de journal simulé (5000 membres, 10 semaines, deux populations) ;
4. calcul : entonnoir, activation (définition écrite), rétention par cohorte hebdomadaire, DAU/MAU, churn ;
5. un cas de **paradoxe de Simpson** que tu construis dans tes données, repéré par ton propre code (détecteur : un test qui compare le sens de l'effet agrégé et par segment) ;
6. enquête de satisfaction simulée : NPS avec intervalle par segment, et comparaison de deux versions avec test (le NPS a-t-il vraiment changé ?) ;

Mini-boss

Mini-boss (suite)

7. 40 retours qualitatifs simulés : classement thématique, comptage de personnes, croisement avec les segments ;
8. une page « tableau de bord » sous forme de texte : chaque indicateur relié à une décision, avec seuil d'alerte.

Mini-boss · réponse

Pistes de correction

Un bon dossier : tableau décisions → questions → métriques → événements, validateur qui rejette les violations (y compris PII), cohortes et DAU/MAU, détecteur de Simpson (fonction qui signale quand le signe de la différence agrégée diffère du signe dans tous les segments), NPS avec intervalle et test de différence, thèmes qualitatifs comptés en personnes, tableau de bord avec seuils d'alerte et décision associée.

Quiz éclair

Quiz éclair

1. Pseudonymiser un identifiant, c'est :

A. le supprimer définitivement et sans retour
B. le publier
C. le remplacer par un pseudonyme dérivé avec une clé secrète, sans être une anonymisation parfaite

2. Le paradoxe de Simpson désigne :

A. une tendance présente dans chaque sous-groupe qui s'inverse dans l'ensemble
B. une erreur de calcul de moyenne
C. une panne de serveur

3. Le NPS se calcule comme :

Quiz éclair

Quiz éclair (suite)

A. moyenne des notes
B. % de promoteurs (9-10) − % de détracteurs (0-6)
C. nombre de réponses

4. 🔁 *Révision mission 07* : Dans RICE, la confiance sert à :

A. augmenter l'effort
B. mesurer la portée
C. pénaliser les estimations spéculatives

5. 🔁 *Révision mission 07* : La règle de Smith (WSJF) ordonne les travaux par :

A. coût du retard par unité de durée, décroissant
B. durée croissante uniquement

Quiz éclair

Quiz éclair (suite)

C. ordre alphabétique

Quiz éclair · réponse

Réponses

1. **C**.
2. **A**.
3. **B**.
4. **C**.
5. **A**.

Je réfléchis

Je réfléchis

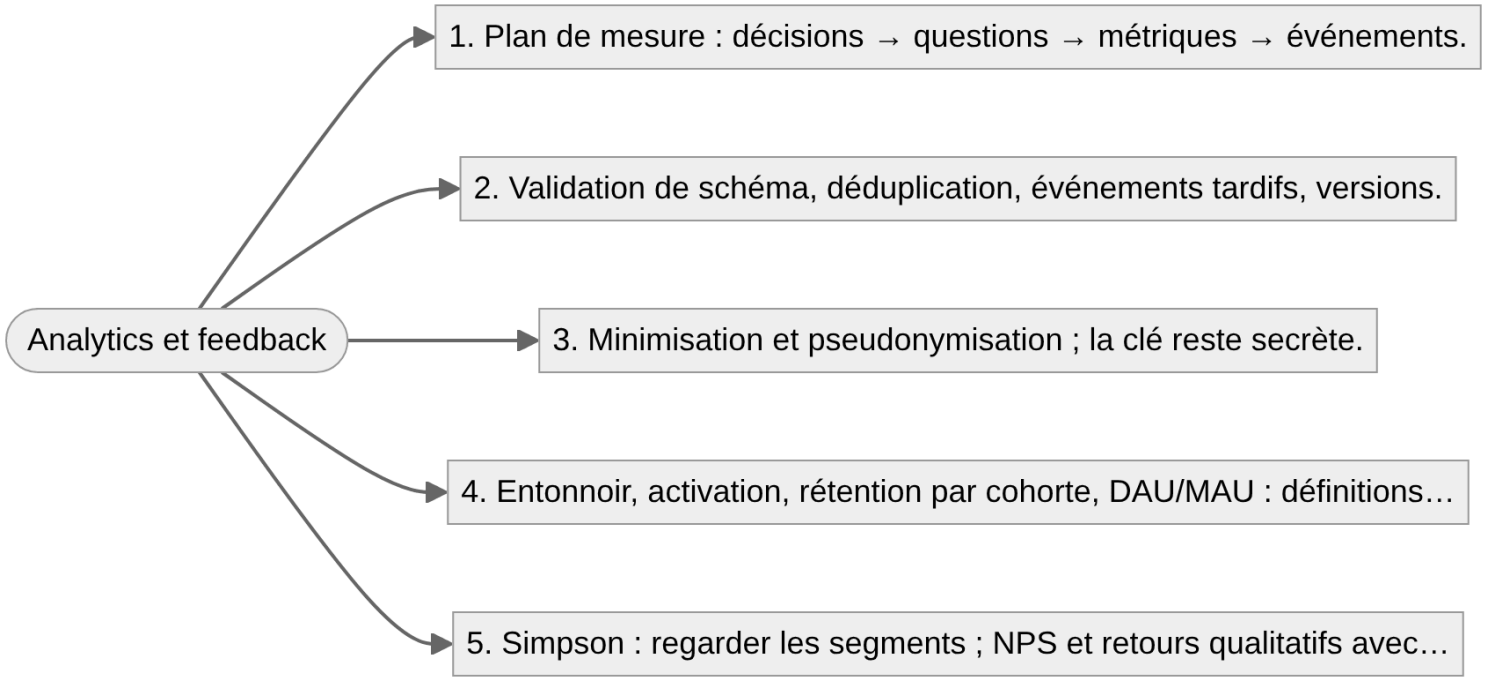
1. Quel chiffre de ton tableau de bord ne changerait aucune décision si tu le supprimais ?
2. Collectes-tu aujourd'hui une donnée personnelle dont tu n'as pas besoin ?

À retenir

1. Plan de mesure : décisions → questions → métriques → événements.
2. Validation de schéma, déduplication, événements tardifs, versions.
3. Minimisation et pseudonymisation ; la clé reste secrète.
4. Entonnoir, activation, rétention par cohorte, DAU/MAU : définitions écrites.
5. Simpson : regarder les segments ; NPS et retours qualitatifs avec incertitude.

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Lecteur de signaux

Prochaine mission : Qualité, dette et delivery

