Analyse_Univariée

September 11, 2025

1 Phase 4 : Analyse, Visualisation et Modélisation

Bienvenue dans ce second notebook dédié à l'analyse approfondie de nos données sur la prévention des MST dans le Borgou.

Objectif : 1. Explorer chaque variable en détail (nettoyage, recodage). 2. Analyser les relations entre le niveau d'instruction et les comportements de prévention. 3. Construire un modèle statistique pour identifier les facteurs les plus influents.

1.1 4.1. Préparation de l'Environnement

Nous commençons par importer les bibliothèques et charger notre jeu de données nettoyé.

[9]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Configuration pour un meilleur affichage
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (12, 7)

print("Bibliothèques importées.")
```

```
Bibliothèques importées.
```

[10]:
```
# Charger le jeu de données
file_path = 'df_analyse_borgou.csv'

try:
    df = pd.read_csv(file_path)
    print(f"Le fichier '{file_path}' a été chargé avec succès.")
    print(f"Dimensions : {df.shape[0]} lignes et {df.shape[1]} colonnes.")
except FileNotFoundError:
    print(f"Erreur : Le fichier '{file_path}' n'a pas été trouvé.")
```

```
Le fichier 'df_analyse_borgou.csv' a été chargé avec succès.
Dimensions : 880 lignes et 19 colonnes.
```

1.2 4.2. Inspection Initiale du DataFrame

Vérifions que les données ont été correctement chargées.

[11]:
```
# Afficher les informations et les premières lignes
if 'df' in locals():
    print("Informations sur le DataFrame :")
    df.info()

    print("\nAperçu des données :")
    display(df.head())
```

```
Informations sur le DataFrame :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 880 entries, 0 to 879
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   region                          880 non-null    float64
 1   usage_preservatif               638 non-null    float64
 2   connaissance_preservatif_vih    864 non-null    float64
 3   connaissance_transmission_sain  864 non-null    float64
 4   a_eu_ist_12mois                 880 non-null    float64
 5   deja_teste_vih                  880 non-null    float64
 6   niveau_instruction              880 non-null    float64
 7   annees_education                880 non-null    float64
 8   alphabetisation                 823 non-null    float64
 9   age                             880 non-null    float64
 10  milieu_residence                880 non-null    float64
 11  statut_marital                  880 non-null    float64
 12  indice_richesse                 880 non-null    float64
 13  travaille_actuellement          880 non-null    float64
 14  frequence_radio                 880 non-null    float64
 15  frequence_tv                    880 non-null    float64
 16  utilise_internet                880 non-null    float64
 17  age_premier_rapport             880 non-null    float64
 18  nb_partenaires_12mois           880 non-null    float64
dtypes: float64(19)
memory usage: 130.8 KB

Aperçu des données :

   region  usage_preservatif  connaissance_preservatif_vih  \
0     4.0                0.0                           1.0
1     4.0                0.0                           1.0
2     4.0                0.0                           1.0
3     4.0                NaN                           1.0
4     4.0                0.0                           1.0
```

```
   connaissance_transmission_sain  a_eu_ist_12mois  deja_teste_vih  \
0                             1.0              0.0             1.0
1                             1.0              0.0             0.0
2                             8.0              0.0             0.0
3                             0.0              0.0             0.0
4                             1.0              0.0             0.0

   niveau_instruction  annees_education  alphabetisation   age  \
0                 0.0               0.0              3.0  27.0
1                 0.0               0.0              0.0  20.0
2                 0.0               0.0              0.0  23.0
3                 0.0               0.0              0.0  15.0
4                 0.0               0.0              0.0  40.0

   milieu_residence  statut_marital  indice_richesse  travaille_actuellement  \
0               2.0             1.0              2.0                     1.0
1               2.0             1.0              2.0                     1.0
2               2.0             1.0              2.0                     1.0
3               2.0             0.0              1.0                     1.0
4               2.0             1.0              1.0                     1.0

   frequence_radio  frequence_tv  utilise_internet  age_premier_rapport  \
0              0.0           0.0               0.0                 19.0
1              2.0           1.0               0.0                 19.0
2              2.0           0.0               0.0                 19.0
3              2.0           1.0               0.0                 13.0
4              0.0           0.0               0.0                 19.0

   nb_partenaires_12mois
0                    1.0
1                    1.0
2                    3.0
3                    0.0
4                    1.0
```

2 Phase 5 : Analyse Exploratoire et Nettoyage des Variables

Maintenant que nos données sont chargées, nous allons examiner chaque variable une par une. Pour un projet de statistique appliquée, cette étape est cruciale pour garantir la qualité de l'analyse finale.

Pour chaque variable, nous suivrons 3 étapes : 1. **Inspection** : Analyser les valeurs brutes et les données manquantes. 2. **Nettoyage/Recodage** : Transformer les données pour les rendre interprétables. 3. **Visualisation** : Représenter la distribution de la variable nettoyée.

2.1 5.1. Analyse des Variables Dépendantes

Commençons par le cœur de notre étude : les variables qui mesurent la prévention.

2.1.1 5.1.1. Variable : usage_preservatif

Cette variable indique si un préservatif a été utilisé lors du dernier rapport sexuel. C'est l'un de nos indicateurs de comportement les plus importants.

Étape 1 : Inspection

Examinons les valeurs brutes de la colonne pour comprendre son codage et identifier les données manquantes ou non applicables.

[12]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'usage_preservatif' :")
    display(df['usage_preservatif'].value_counts(dropna=False))
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'usage_preservatif' :

usage_preservatif
0.0    550
NaN    242
1.0     88
Name: count, dtype: int64
```

[13]:
```
# Pour interpréter les codes numériques, nous devons récupérer les métadonnées
# du fichier .sav original.
import pyreadstat

sav_file_path = 'BJMR71DT/BJMR71DT.sav'
try:
    _, meta = pyreadstat.read_sav(sav_file_path)
    print("Métadonnées chargées avec succès.")
except FileNotFoundError:
    print(f"Erreur : Le fichier '{sav_file_path}' est nécessaire pour les␣
 ↪libellés.")

# Afficher les libellés pour la variable 'usage_preservatif' (nom original :␣
 ↪mv761)
if 'meta' in locals():
    variable_name = 'mv761'
    if variable_name in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name}':")
        for value, label in meta.variable_value_labels[variable_name].items():
            print(f"  {int(value)}: {label}")
    else:
        print(f"Pas de libellés trouvés pour '{variable_name}'.")
```

```
Métadonnées chargées avec succès.

Libellés pour 'mv761':
```

```
0: no
1: yes
8: don't know
```

Étape 1 (suite) : Interprétation

L'inspection révèle trois catégories dans notre échantillon du Borgou :

- **1.0 (Oui)** : 88 répondants
- **0.0 (Non)** : 550 répondants
- **NaN (Manquant)** : 242 répondants

Analyse Statistique
La présence de **242 valeurs manquantes (NaN)** est significative. Dans le contexte des enquêtes EDS, cela correspond généralement à un filtre dans le questionnaire : la question sur l'usage du préservatif n'est posée qu'aux hommes ayant eu une activité sexuelle dans une période de référence récente.

Stratégie de Traitement
Pour analyser le comportement (usage ou non du préservatif), notre population d'intérêt est constituée uniquement des hommes ayant eu une activité sexuelle et ayant répondu à la question. Les **242 cas NaN** seront donc exclus de l'analyse spécifique à cette variable.

Nous allons maintenant recoder la variable pour la rendre plus lisible.

Étape 2 : Nettoyage et Recodage

Nous allons créer une nouvelle colonne avec des libellés clairs pour faciliter les visualisations et les interprétations.

[14]:
```
if 'df' in locals():
    # Définir le mapping
    mapping = {
        0.0: 'Non',
        1.0: 'Oui',
        np.nan: 'Non concerné / Manquant'
    }

    # Appliquer le mapping pour créer une nouvelle colonne de visualisation
    df['usage_preservatif_cat'] = df['usage_preservatif'].map(mapping)

    print("Distribution de la variable recodée :")
    display(df['usage_preservatif_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

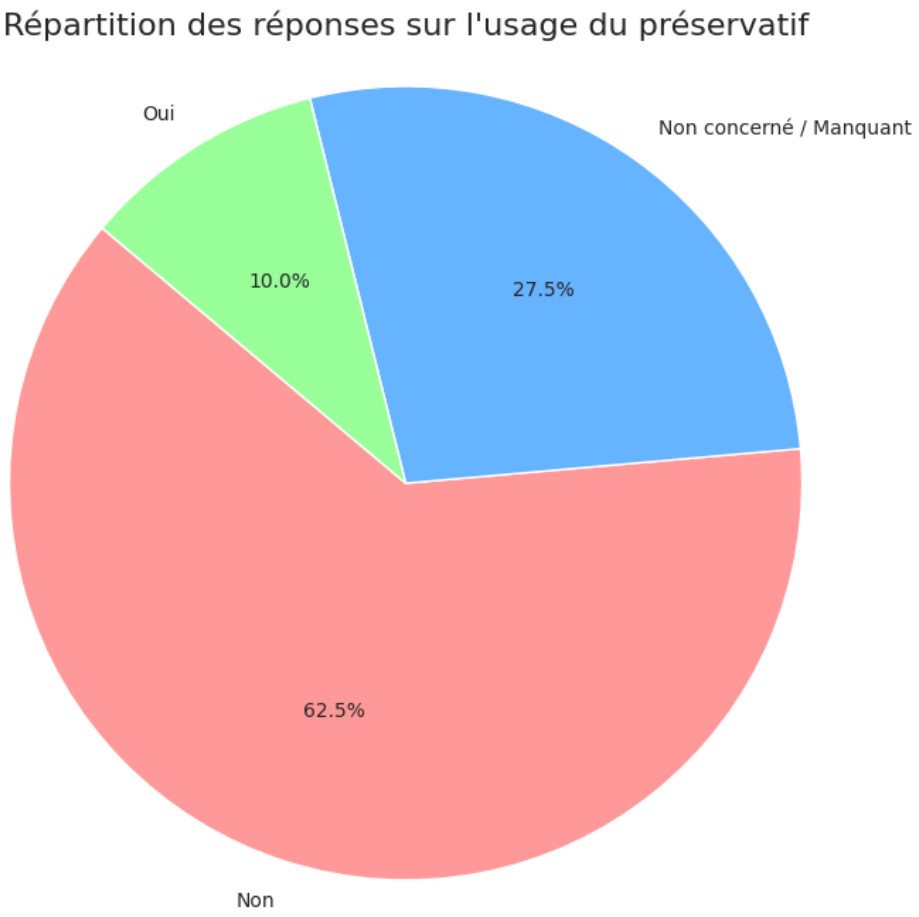
```
Distribution de la variable recodée :

usage_preservatif_cat
Non                     550
```

```
Non concerné / Manquant    242
Oui                         88
Name: count, dtype: int64
```

Étape 3 : Visualisation

Un diagramme circulaire (camembert) est parfait pour visualiser la proportion de chaque catégorie.

[15]:
```
if 'df' in locals():
    # Calculer les effectifs
    counts = df['usage_preservatif_cat'].value_counts()

    # Créer le diagramme circulaire
    plt.figure(figsize=(10, 8))
    plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=140,␣
 ↪colors=['#ff9999','#66b3ff','#99ff99'])

    # Ajouter un titre
    plt.title('Répartition des réponses sur l\'usage du préservatif',␣
 ↪fontsize=16)

    # Assurer un cercle parfait
    plt.axis('equal')

    # Afficher le graphique
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```



2.1.2 5.1.2. Variable : `connaissance_preservatif_vih`

Cette variable mesure si le répondant sait que l'utilisation systématique du préservatif peut réduire le risque de contracter le VIH. C'est un indicateur de connaissance fondamental.

Étape 1 : Inspection

[16]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'connaissance_preservatif_vih' :
 ↪")
    display(df['connaissance_preservatif_vih'].value_counts(dropna=False))

    # Afficher les libellés correspondants
    variable_name_orig = 'mv754cp'
    if 'meta' in locals() and variable_name_orig in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name_orig}':")
```

```
        for value, label in meta.variable_value_labels[variable_name_orig].
 ↪items():
            print(f"  {int(value)}: {label}")
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'connaissance_preservatif_vih' :

connaissance_preservatif_vih
1.0    719
0.0     77
8.0     68
NaN     16
Name: count, dtype: int64


Libellés pour 'mv754cp':
  0: no
  1: yes
  8: don't know
```

Étape 1 (suite) : Interprétation

Les résultats de l'inspection sont les suivants : - `1.0` **(Oui)** : 719 répondants. - `0.0` **(Non)** : 77 répondants. - `8.0` **(Ne sait pas)** : 68 répondants. - `NaN` **(Manquant)** : 16 répondants.

Analyse Statistique : Le niveau de connaissance est globalement très élevé, avec une écrasante majorité (719 sur 880 répondants valides) qui connaît le rôle préventif du préservatif. Cependant, les catégories "Non" et "Ne sait pas" ne sont pas négligeables et représentent des cibles potentielles pour des campagnes de sensibilisation. Les 16 valeurs manquantes seront exclues des analyses sur cette variable.

Stratégie de Traitement : Nous allons recoder la variable en quatre catégories claires pour l'analyse et la visualisation.

Étape 2 : Nettoyage et Recodage

[17]:
```
if 'df' in locals():
    # Définir le mapping
    mapping_connaissance = {
        1.0: 'Oui',
        0.0: 'Non',
        8.0: 'Ne sait pas',
        np.nan: 'Manquant'
    }

    # Appliquer le mapping
    df['connaissance_preservatif_vih_cat'] = df['connaissance_preservatif_vih'].
 ↪map(mapping_connaissance)

    print("Distribution de la variable recodée :")
```

```
    display(df['connaissance_preservatif_vih_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution de la variable recodée :

connaissance_preservatif_vih_cat
Oui            719
Non             77
Ne sait pas     68
Manquant        16
Name: count, dtype: int64
```

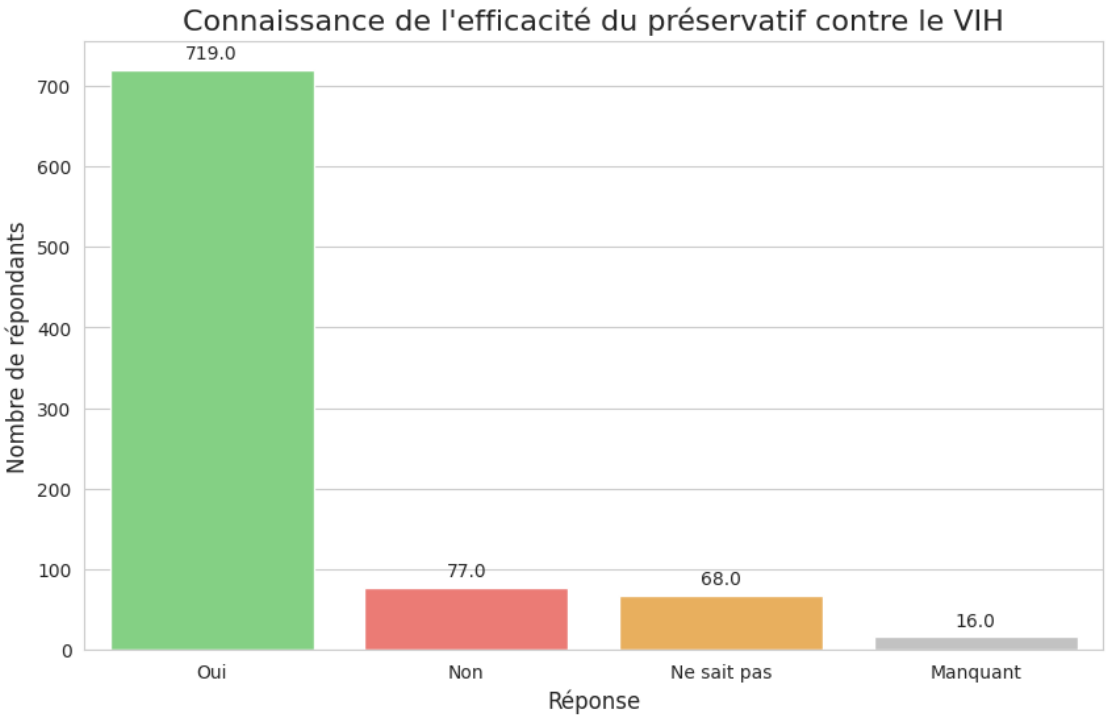
Étape 3 : Visualisation

Un diagramme en barres est idéal pour comparer les effectifs de chaque catégorie de réponse.

[18]:
```
if 'df' in locals():
    # Créer le diagramme en barres
    plt.figure(figsize=(10, 6))
    ax = sns.countplot(x='connaissance_preservatif_vih_cat',
                       data=df,
                       order=['Oui', 'Non', 'Ne sait pas', 'Manquant'],
                       palette=['#77dd77','#ff6961','#ffb347', '#c2c2c2'])

    # Ajouter les effectifs au-dessus des barres
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center',
                    xytext=(0, 9),
                    textcoords='offset points')

    # Titres et labels
    plt.title('Connaissance de l\'efficacité du préservatif contre le VIH',␣
 ↪fontsize=16)
    plt.xlabel('Réponse', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)

    # Afficher le graphique
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/1927648325.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```

```
ax = sns.countplot(x='connaissance_preservatif_vih_cat',
```



2.1.3 5.1.3. Variable : `connaissance_transmission_sain`

Cette variable évalue si le répondant est conscient qu'une personne d'apparence saine peut être porteuse du VIH. C'est une mesure de la connaissance sur la transmission asymptomatique.

Étape 1 : Inspection

[19]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour␣
 ↪'connaissance_transmission_sain' :")
    display(df['connaissance_transmission_sain'].value_counts(dropna=False))

    # Afficher les libellés correspondants
    variable_name_orig = 'mv756'
    if 'meta' in locals() and variable_name_orig in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name_orig}':")
        for value, label in meta.variable_value_labels[variable_name_orig].
 ↪items():
            print(f"  {int(value)}: {label}")
else:
```

```
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'connaissance_transmission_sain' :

connaissance_transmission_sain
1.0    711
0.0     88
8.0     65
NaN     16
Name: count, dtype: int64


Libellés pour 'mv756':
  0: no
  1: yes
  8: don't know
```

Étape 1 (suite) : Interprétation

Les résultats pour cette variable sont très similaires à la précédente : - `1.0` **(Oui)** : 711 répondants. - `0.0` **(Non)** : 88 répondants. - `8.0` **(Ne sait pas)** : 65 répondants. - `NaN` **(Manquant)** : 16 répondants.

Analyse Statistique : Encore une fois, une forte majorité de la population interrogée est consciente du risque de transmission par une personne d'apparence saine. Les groupes "Non" et "Ne sait pas" restent des cibles pertinentes pour des actions de sensibilisation. Les 16 valeurs manquantes seront écartées de l'analyse de cette variable.

Stratégie de Traitement : Nous allons recoder la variable en quatre catégories claires : 'Oui', 'Non', 'Ne sait pas', et 'Manquant'.

Étape 2 : Nettoyage et Recodage

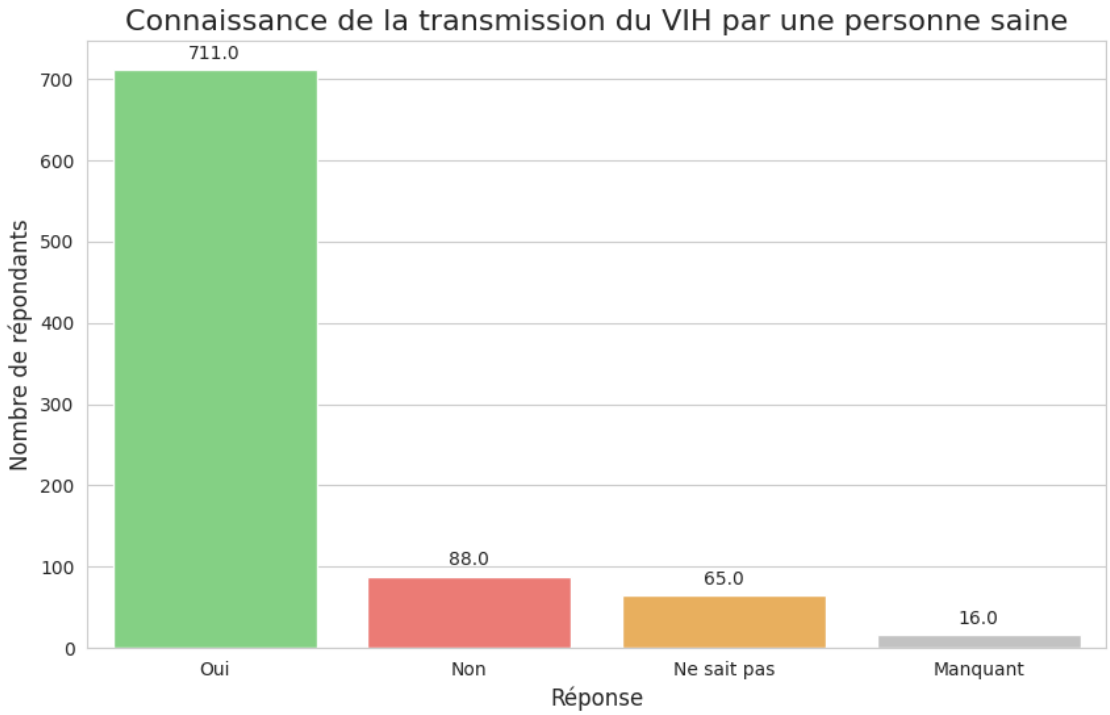
[20]:
```
if 'df' in locals():
    # Définir le mapping
    mapping_sain = {
        1.0: 'Oui',
        0.0: 'Non',
        8.0: 'Ne sait pas',
        np.nan: 'Manquant'
    }

    # Appliquer le mapping
    df['connaissance_transmission_sain_cat'] =␣
 ↪df['connaissance_transmission_sain'].map(mapping_sain)

    print("Distribution de la variable recodée :")
    display(df['connaissance_transmission_sain_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution de la variable recodée :

connaissance_transmission_sain_cat
Oui            711
Non             88
Ne sait pas     65
Manquant        16
Name: count, dtype: int64
```

Étape 3 : Visualisation

[21]:
```python
if 'df' in locals():
    # Créer le diagramme en barres
    plt.figure(figsize=(10, 6))
    ax = sns.countplot(x='connaissance_transmission_sain_cat',
                       data=df,
                       order=['Oui', 'Non', 'Ne sait pas', 'Manquant'],
                       palette=['#77dd77','#ff6961','#ffb347', '#c2c2c2'])

    # Ajouter les effectifs au-dessus des barres
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center',
                    xytext=(0, 9),
                    textcoords='offset points')

    # Titres et labels
    plt.title('Connaissance de la transmission du VIH par une personne saine',␣
 ↪fontsize=16)
    plt.xlabel('Réponse', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)

    # Afficher le graphique
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/985136587.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x='connaissance_transmission_sain_cat',
```



2.1.4 5.1.4. Variable : `a_eu_ist_12mois`

Cette variable de perception du risque indique si le répondant a eu une Infection Sexuellement Transmissible (IST) ou des symptômes d'IST au cours des 12 derniers mois.

Étape 1 : Inspection

[22]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'a_eu_ist_12mois' :")
    display(df['a_eu_ist_12mois'].value_counts(dropna=False))

    # Afficher les libellés correspondants
    variable_name_orig = 'mv763a'
    if 'meta' in locals() and variable_name_orig in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name_orig}':")
        for value, label in meta.variable_value_labels[variable_name_orig].
 ↪items():
            print(f"  {int(value)}: {label}")
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'a_eu_ist_12mois' :

a_eu_ist_12mois
```

```
0.0    865
1.0     14
8.0      1
Name: count, dtype: int64


Libellés pour 'mv763a':
  0: no
  1: yes
  8: don't know
```

Étape 1 (suite) : Interprétation

Les résultats de l'inspection sont les suivants : - `0.0` **(Non)** : 865 répondants. - `1.0` **(Oui)** : 14 répondants. - `8.0` **(Ne sait pas)** : 1 répondant. - `NaN` **(Manquant)** : 0.

Analyse Statistique : Une très faible proportion des hommes interrogés (14 sur 880, soit environ 1,6%) déclare avoir eu une IST ou des symptômes au cours des 12 derniers mois. La quasi-totalité (98,3%) n'en a pas eu. Le fait qu'il n'y ait aucune valeur manquante suggère que la question a été posée à l'ensemble de l'échantillon.

Stratégie de Traitement : Nous allons recoder la variable en trois catégories claires : 'Oui', 'Non', et 'Ne sait pas'. La catégorie 'Ne sait pas' est très marginale (1 seul cas) mais nous la conserverons pour l'instant.

Étape 2 : Nettoyage et Recodage

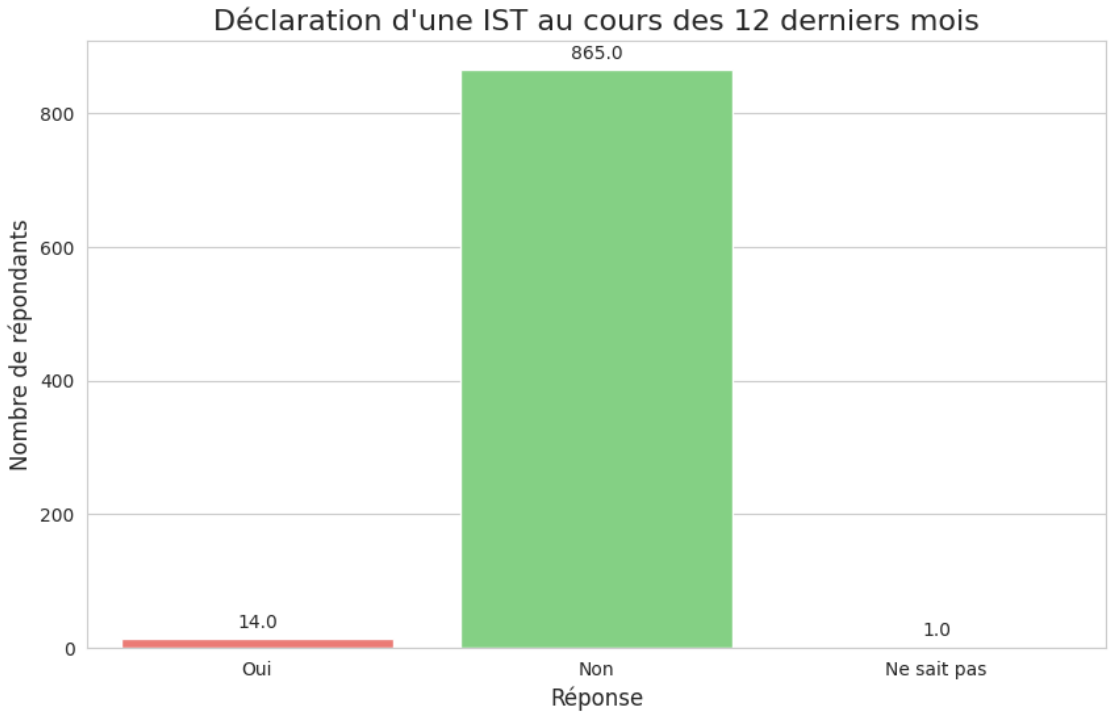
[23]:
```python
if 'df' in locals():
    # Définir le mapping
    mapping_ist = {
        1.0: 'Oui',
        0.0: 'Non',
        8.0: 'Ne sait pas',
        np.nan: 'Manquant' # Même si absent, c'est une bonne pratique
    }

    # Appliquer le mapping
    df['a_eu_ist_12mois_cat'] = df['a_eu_ist_12mois'].map(mapping_ist)

    print("Distribution de la variable recodée :")
    display(df['a_eu_ist_12mois_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution de la variable recodée :

a_eu_ist_12mois_cat
Non            865
Oui             14
Ne sait pas      1
Name: count, dtype: int64
```

Étape 3 : Visualisation

[24]:
```
if 'df' in locals():
    # Créer le diagramme en barres
    plt.figure(figsize=(10, 6))
    ax = sns.countplot(x='a_eu_ist_12mois_cat',
                       data=df,
                       order=['Oui', 'Non', 'Ne sait pas'],
                       palette=['#ff6961', '#77dd77', '#ffb347'])

    # Ajouter les effectifs au-dessus des barres
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center',
                    xytext=(0, 9),
                    textcoords='offset points')

    # Titres et labels
    plt.title('Déclaration d\'une IST au cours des 12 derniers mois',␣
 ↪fontsize=16)
    plt.xlabel('Réponse', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)

    # Afficher le graphique
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/1024020907.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x='a_eu_ist_12mois_cat',
```



2.1.5 5.1.5. Variable : `deja_teste_vih`

Cette variable indique si le répondant a déjà fait un test de dépistage du VIH. C'est un indicateur clé du recours aux services de santé préventive.

Étape 1 : Inspection

[25]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'deja_teste_vih' :")
    display(df['deja_teste_vih'].value_counts(dropna=False))

    # Afficher les libellés correspondants
    variable_name_orig = 'mv781'
    if 'meta' in locals() and variable_name_orig in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name_orig}':")
        for value, label in meta.variable_value_labels[variable_name_orig].
 ↪items():
            print(f"  {int(value)}: {label}")
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'deja_teste_vih' :

deja_teste_vih
```

```
0.0    771
1.0    109
Name: count, dtype: int64


Libellés pour 'mv781':
  0: no
  1: yes
```

Étape 1 (suite) : Interprétation

Les résultats de l'inspection sont très clairs : - `0.0` **(Non)** : 771 répondants. - `1.0` **(Oui)** : 109 répondants. - `NaN` **(Manquant)** : 0.

Analyse Statistique : Une minorité significative (109 sur 880, soit environ 12,4%) des hommes interrogés dans le Borgou a déjà effectué un test de dépistage du VIH. La grande majorité (87,6%) ne l'a jamais fait. L'absence de valeurs manquantes indique que la question a été posée à tous les participants de notre échantillon.

Stratégie de Traitement : La variable est déjà propre. Nous allons simplement la recoder avec des libellés explicites pour l'analyse.

Étape 2 : Nettoyage et Recodage

[26]:
```
if 'df' in locals():
    # Définir le mapping
    mapping_test = {
        1.0: 'Oui',
        0.0: 'Non',
        np.nan: 'Manquant'
    }

    # Appliquer le mapping
    df['deja_teste_vih_cat'] = df['deja_teste_vih'].map(mapping_test)

    print("Distribution de la variable recodée :")
    display(df['deja_teste_vih_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

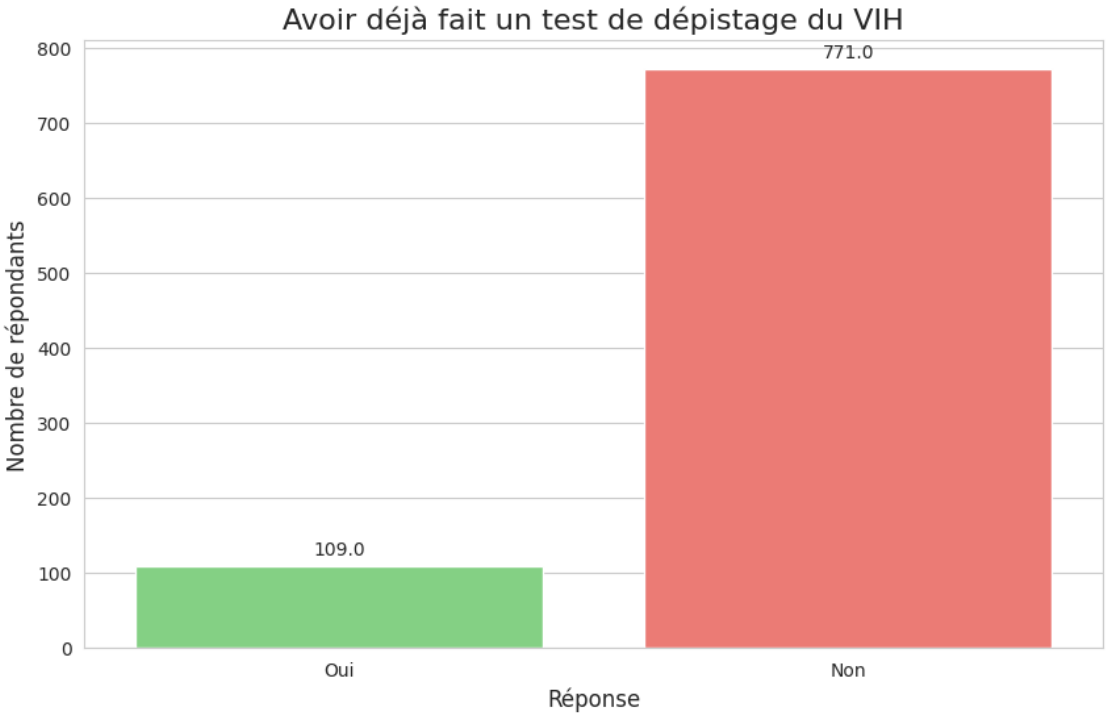
```
Distribution de la variable recodée :

deja_teste_vih_cat
Non    771
Oui    109
Name: count, dtype: int64
```

Étape 3 : Visualisation

[29]:
```
if 'df' in locals():
    # Créer le diagramme en barres
    plt.figure(figsize=(10, 6))
```

```
    ax = sns.countplot(x='deja_teste_vih_cat',
                       data=df,
                       order=['Oui', 'Non'],
                       palette=['#77dd77', '#ff6961'])

    # Ajouter les effectifs au-dessus des barres
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center',
                    xytext=(0, 9),
                    textcoords='offset points')

    # Titres et labels
    plt.title('Avoir déjà fait un test de dépistage du VIH', fontsize=16)
    plt.xlabel('Réponse', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)

    # Afficher le graphique
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/1628593959.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x='deja_teste_vih_cat',
```



2.1.6 5.1.6. Synthèse de l'analyse des variables dépendantes

Nous avons terminé l'exploration de nos cinq variables dépendantes. Voici un résumé de nos observations :

1. **`usage_preservatif`** : Une part importante de l'échantillon n'a pas utilisé de préservatif lors du dernier rapport, et un grand nombre n'était pas concerné par la question, probablement en raison de l'inactivité sexuelle récente.
2. **`connaissance_preservatif_vih`** : Le niveau de connaissance sur l'efficacité du préservatif est très élevé.
3. **`connaissance_transmission_sain`** : De même, la connaissance sur la transmission asymptomatique du VIH est très répandue.
4. **`a_eu_ist_12mois`** : La prévalence déclarée d'IST au cours de la dernière année est très faible dans notre échantillon.
5. **`deja_teste_vih`** : Le recours au dépistage du VIH reste minoritaire, avec une large majorité n'ayant jamais fait le test.

Cette première analyse nous donne une image claire des comportements et des connaissances en matière de santé sexuelle dans notre population d'étude. Nous sommes maintenant prêts à explorer les facteurs qui pourraient influencer ces variables.

2.2 5.2. Analyse des Variables Indépendantes

Nous passons maintenant à l'analyse des variables indépendantes, qui sont au cœur de notre problématique. Celles-ci concernent directement le niveau d'éducation des répondants.

2.2.1 5.2.1. Variable : `niveau_instruction`

Le niveau d'instruction est notre principal facteur explicatif. Nous allons l'analyser en détail.

Étape 1 : Inspection

[27]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'niveau_instruction' :")
    display(df['niveau_instruction'].value_counts(dropna=False))

    # Afficher les libellés correspondants
    variable_name_orig = 'mv106'
    if 'meta' in locals() and variable_name_orig in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name_orig}':")
        for value, label in meta.variable_value_labels[variable_name_orig].
 ↪items():
            print(f"  {int(value)}: {label}")
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'niveau_instruction' :

niveau_instruction
0.0    466
2.0    208
1.0    149
3.0     57
Name: count, dtype: int64


Libellés pour 'mv106':
  0: no education
  1: primary
  2: secondary
  3: higher
```

Étape 1 (suite) : Interprétation

Les résultats de l'inspection sont les suivants : - `0.0` **(Aucune éducation)** : 466 répondants. - `1.0` **(Primaire)** : 149 répondants. - `2.0` **(Secondaire)** : 208 répondants. - `3.0` **(Supérieur)** : 57 répondants. - `NaN` **(Manquant)** : 0.

Analyse Statistique : Le niveau d'instruction dans notre échantillon est très polarisé. Une majorité significative (466 sur 880, soit 53%) n'a reçu aucune éducation formelle. Le niveau secondaire représente le deuxième groupe le plus important (23,6%), suivi du primaire (16,9%) et du supérieur (6,5%). L'absence de valeurs manquantes est une excellente nouvelle pour la modélisation.

Stratégie de Traitement : La variable est propre et ordinale. Nous allons la recoder avec des libellés clairs pour l'analyse et la visualisation.

Étape 2 : Nettoyage et Recodage

[28]:
```
if 'df' in locals():
    # Définir le mapping
    mapping_instruction = {
        0.0: 'Aucune',
        1.0: 'Primaire',
        2.0: 'Secondaire',
        3.0: 'Supérieur',
        np.nan: 'Manquant'
    }

    # Appliquer le mapping
    df['niveau_instruction_cat'] = df['niveau_instruction'].
 ↪map(mapping_instruction)

    print("Distribution de la variable recodée :")
    display(df['niveau_instruction_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

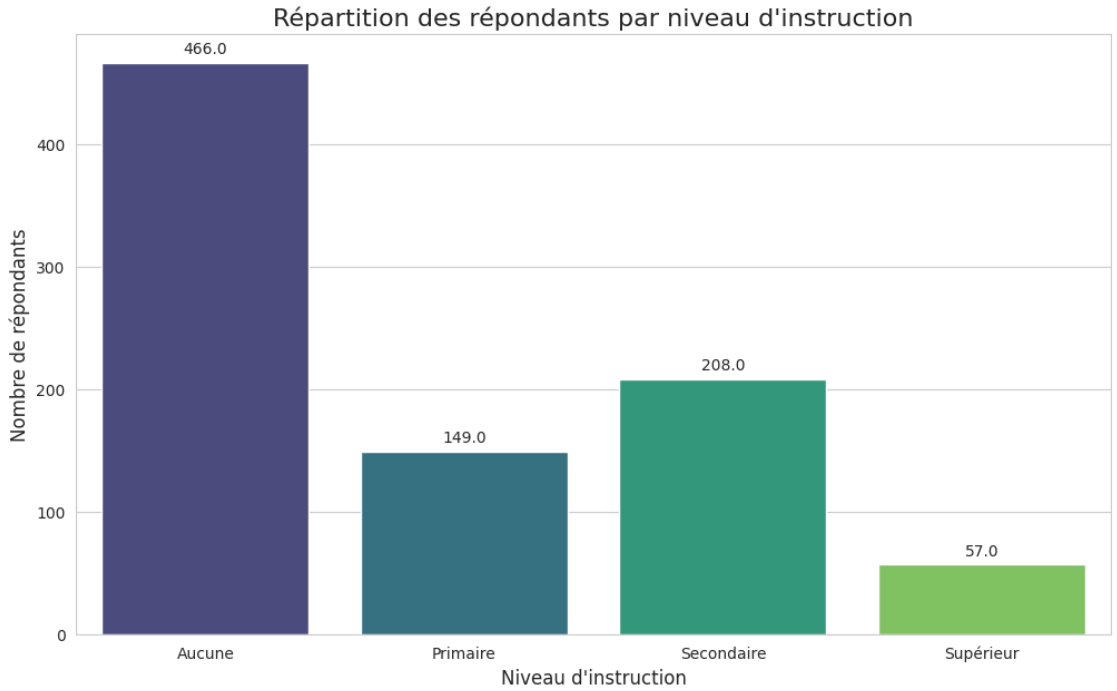
```
Distribution de la variable recodée :

niveau_instruction_cat
Aucune        466
Secondaire    208
Primaire      149
Supérieur      57
Name: count, dtype: int64
```

Étape 3 : Visualisation

[30]:
```
if 'df' in locals():
    # Définir l'ordre des catégories
    order = ['Aucune', 'Primaire', 'Secondaire', 'Supérieur']

    # Créer le diagramme en barres
    plt.figure(figsize=(12, 7))
    ax = sns.countplot(x='niveau_instruction_cat',
                       data=df,
                       order=order,
                       palette='viridis')

    # Ajouter les effectifs au-dessus des barres
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}',
```

```
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center',
                    xytext=(0, 9),
                    textcoords='offset points')

    # Titres et labels
    plt.title('Répartition des répondants par niveau d\'instruction',␣
 ↪fontsize=16)
    plt.xlabel('Niveau d\'instruction', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)

    # Afficher le graphique
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/3662687991.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x='niveau_instruction_cat',
```



2.2.2 5.2.2. Variable : annees_education

Cette variable numérique complète la précédente en donnant le nombre d'années d'études.

Étape 1 : Inspection

[36]:
```
if 'df' in locals():
    print("Description statistique de 'annees_education' :")
    display(df['annees_education'].describe())

    print("\nDistribution des fréquences pour 'annees_education' :")
    display(df['annees_education'].value_counts(dropna=False).sort_index())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Description statistique de 'annees_education' :
```

```
count    880.000000
mean       4.014773
std        5.570605
min        0.000000
25%        0.000000
50%        0.000000
75%        7.000000
max       21.000000
Name: annees_education, dtype: float64
```

```
Distribution des fréquences pour 'annees_education' :
```

```
annees_education
0.0     469
1.0      10
2.0      31
3.0      25
4.0      19
5.0      45
6.0      28
7.0      36
8.0      40
9.0      46
10.0     21
11.0     17
12.0     27
13.0      9
17.0      5
18.0     15
19.0     13
20.0     13
21.0     11
Name: count, dtype: int64
```

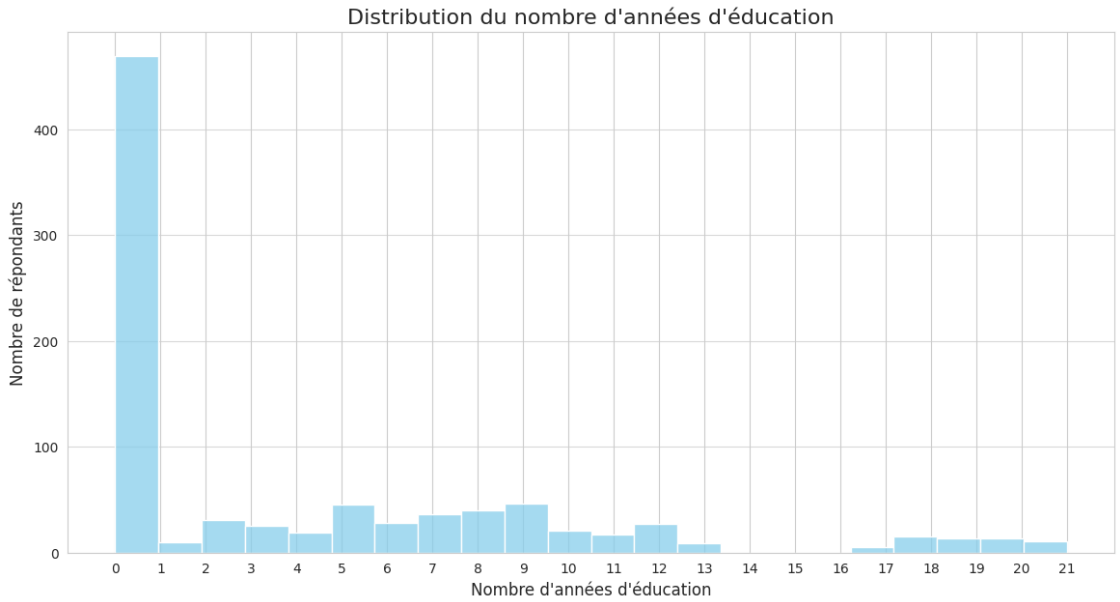
Étape 1 (suite) : Interprétation

L'analyse de la variable `annees_education` confirme ce que nous avons vu avec `niveau_instruction` : - **Moyenne de 4 ans d'études**, mais fortement influencée par les valeurs extrêmes (jusqu'à 21 ans). - **Médiane à 0**, ce qui est très parlant : au moins 50% de notre échantillon n'a aucune année d'éducation formelle. - **Forte concentration à 0 an** (469 personnes), ce qui créera une distribution très asymétrique.

Stratégie de Traitement : Cette variable est numérique et peut être utilisée directement dans certains modèles. Pour la visualisation et l'analyse descriptive, un histogramme nous permettra de bien voir sa distribution.

Étape 2 : Visualisation

[38]:
```
if 'df' in locals():
    plt.figure(figsize=(14, 7))
    sns.histplot(df['annees_education'], bins=22, kde=False, color='skyblue')
    plt.title('Distribution du nombre d\'années d\'éducation', fontsize=16)
    plt.xlabel('Nombre d\'années d\'éducation', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)
    plt.xticks(range(0, 22))
    plt.grid(axis='y', alpha=0.75)
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```



2.2.3 5.2.3. Variable : `alphabetisation`

Cette variable mesure la capacité à lire et écrire, ce qui est une compétence fonctionnelle clé, parfois décorrélée du niveau d'instruction formel.

Étape 1 : Inspection

[37]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'alphabetisation' :")
    display(df['alphabetisation'].value_counts(dropna=False))

    # Afficher les libellés correspondants
    variable_name_orig = 'mv155'
    if 'meta' in locals() and variable_name_orig in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name_orig}':")
        for value, label in meta.variable_value_labels[variable_name_orig].
 ↪items():
            print(f"  {int(value)}: {label}")
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'alphabetisation' :
```

```
alphabetisation
0.0    516
2.0    219
1.0     63
NaN     57
3.0     24
4.0      1
Name: count, dtype: int64


Libellés pour 'mv155':
  0: cannot read at all
  1: able to read only parts of sentence
  2: able to read whole sentence
  3: no card with required language
  4: blind/visually impaired
```

Étape 1 (suite) : Interprétation

L'alphabétisation présente plusieurs catégories : - `0.0` **(Ne sait pas lire)** : 516 personnes (la majorité). - `2.0` **(Sait lire une phrase entière)** : 219 personnes. - `1.0` **(Sait lire une partie de phrase)** : 63 personnes. - `NaN` **(Manquant)** : 57 personnes. - `3.0` **(Pas de carte dans la langue requise)** : 24 personnes. - `4.0` **(Aveugle/Malvoyant)** : 1 personne.

Stratégie de Traitement : Pour simplifier l'analyse, nous allons regrouper ces catégories. - Les catégories `1.0` et `2.0` peuvent être combinées en "Sait lire". - La catégorie `0.0` reste "Ne sait pas lire". - Les catégories `3.0`, `4.0` et `NaN` sont des cas particuliers ou des données manquantes. Nous les regrouperons en "Autre / Manquant".

Étape 2 : Nettoyage et Recodage

[39]:
```
if 'df' in locals():
    # Définir le mapping
    mapping_alpha = {
        0.0: 'Ne sait pas lire',
        1.0: 'Sait lire',
        2.0: 'Sait lire',
        3.0: 'Autre / Manquant',
        4.0: 'Autre / Manquant',
        np.nan: 'Autre / Manquant'
    }

    # Appliquer le mapping
    df['alphabetisation_cat'] = df['alphabetisation'].map(mapping_alpha)

    print("Distribution de la variable recodée :")
    display(df['alphabetisation_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

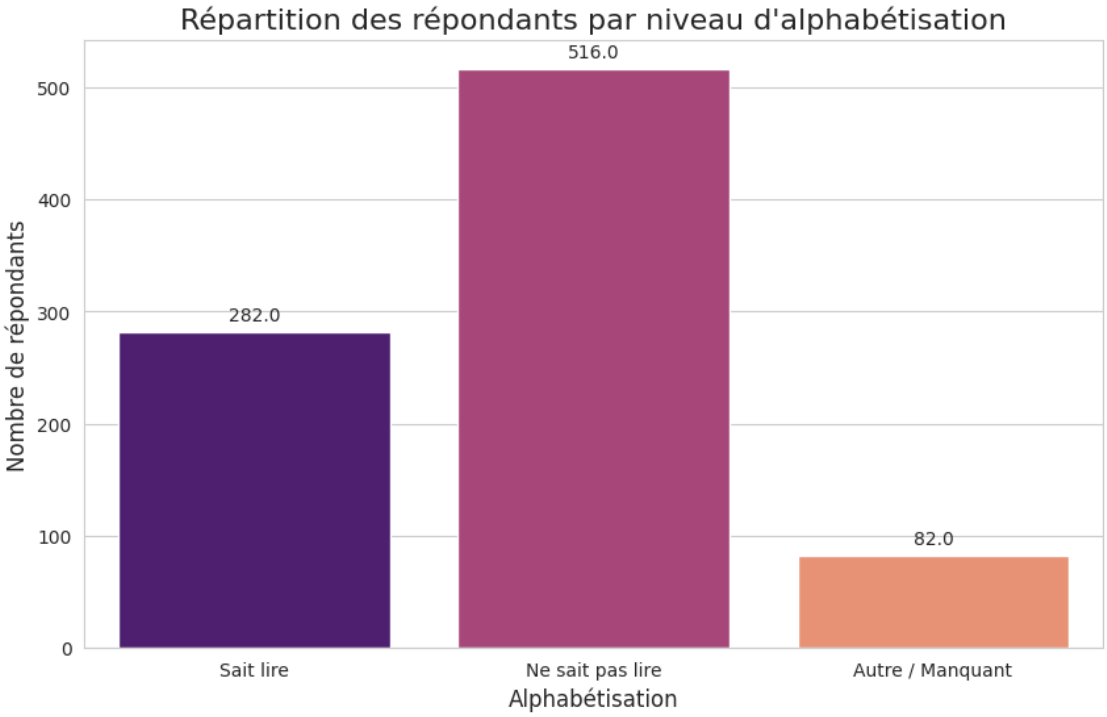
```
Distribution de la variable recodée :

alphabetisation_cat
Ne sait pas lire    516
Sait lire           282
Autre / Manquant     82
Name: count, dtype: int64
```

Étape 3 : Visualisation

[40]:
```
if 'df' in locals():
    order = ['Sait lire', 'Ne sait pas lire', 'Autre / Manquant']
    plt.figure(figsize=(10, 6))
    ax = sns.countplot(x='alphabetisation_cat', data=df, order=order,␣
 ↪palette='magma')

    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                    ha='center', va='center', xytext=(0, 9), textcoords='offset␣
 ↪points')

    plt.title('Répartition des répondants par niveau d\'alphabétisation',␣
 ↪fontsize=16)
    plt.xlabel('Alphabétisation', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)
    plt.show()
else:
```

```
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/2209457728.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x='alphabetisation_cat', data=df, order=order,
palette='magma')
```



2.3 5.3. Analyse des Variables de Contrôle

Nous analysons maintenant les variables socio-démographiques qui ne sont pas directement liées à l'éducation mais qui pourraient influencer la relation entre l'éducation et les comportements de santé.

2.3.1 5.3.1. Variable : age

L'âge est un facteur démographique fondamental.

Étape 1 : Inspection

[35]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'age' :")
    display(df['age'].describe())

    print("\nDistribution des fréquences pour 'age' :")
    display(df['age'].value_counts(dropna=False).sort_index())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'age' :
```

```
count    880.000000
mean      31.445455
std       12.002056
min       15.000000
25%       22.000000
50%       29.000000
75%       40.000000
max       64.000000
Name: age, dtype: float64
```

```
Distribution des fréquences pour 'age' :
```

```
age
15.0    37
16.0    28
17.0    26
18.0    32
19.0    33
20.0    41
21.0    20
22.0    33
23.0    26
24.0    18
25.0    49
26.0    25
27.0    24
28.0    27
29.0    24
30.0    33
31.0    20
32.0    21
33.0    20
34.0    13
35.0    39
36.0    10
37.0    31
38.0    21
```

```
39.0     6
40.0    25
41.0    14
42.0    16
43.0    19
44.0     8
45.0    16
46.0     5
47.0    19
48.0    13
49.0     6
50.0    11
51.0     4
52.0     7
53.0     6
54.0     4
55.0    11
56.0     3
57.0     4
58.0     5
59.0     6
60.0     7
61.0     4
62.0     3
63.0     6
64.0     1
Name: count, dtype: int64
```

Étape 1 (suite) : Interprétation

L'âge des répondants s'étend de 15 à 64 ans, avec une moyenne autour de 31 ans. La distribution est assez large, ce qui justifie de la regrouper en catégories pour l'analyse.

Stratégie de Traitement : Nous allons créer une variable catégorielle `age_cat` en découpant l'âge en tranches. Des groupes standards pour ce type d'enquête sont pertinents : - 15-24 ans (Jeunes) - 25-34 ans (Jeunes Adultes) - 35-49 ans (Adultes) - 50 ans et plus (Adultes plus âgés)

Étape 2 : Nettoyage et Recodage

[41]:
```python
if 'df' in locals():
    # Définir les bornes et les libellés
    bins = [14, 24, 34, 49, 100]
    labels = ['15-24 ans', '25-34 ans', '35-49 ans', '50 ans et plus']

    # Créer la nouvelle variable
    df['age_cat'] = pd.cut(df['age'], bins=bins, labels=labels, right=True)

    print("Distribution de la variable d'âge recodée :")
    display(df['age_cat'].value_counts())
```

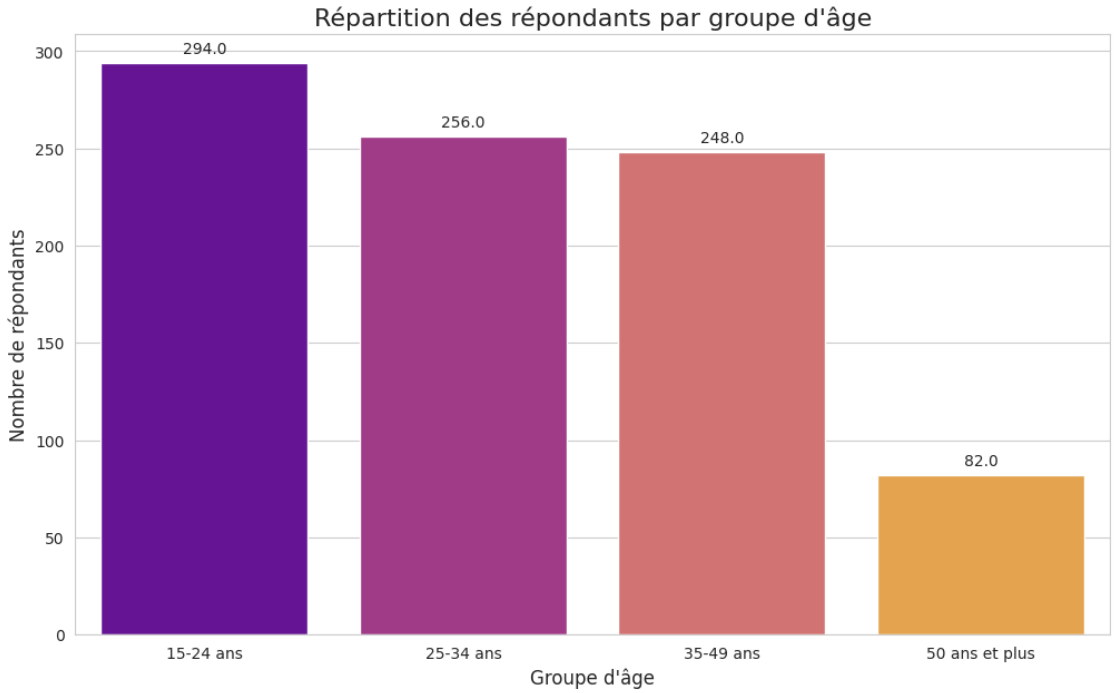
```
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution de la variable d'âge recodée :

age_cat
15-24 ans         294
25-34 ans         256
35-49 ans         248
50 ans et plus     82
Name: count, dtype: int64
```

Étape 3 : Visualisation

[42]:
```
if 'df' in locals():
    order = ['15-24 ans', '25-34 ans', '35-49 ans', '50 ans et plus']
    plt.figure(figsize=(12, 7))
    ax = sns.countplot(x='age_cat', data=df, order=order, palette='plasma')

    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                    ha='center', va='center', xytext=(0, 9), textcoords='offset␣
 ↪points')

    plt.title('Répartition des répondants par groupe d\'âge', fontsize=16)
    plt.xlabel('Groupe d\'âge', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/1867730329.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x='age_cat', data=df, order=order, palette='plasma')
```



2.3.2 5.3.2. Inspection Groupée des Autres Variables de Contrôle

Pour être plus efficaces, nous allons inspecter les variables de contrôle restantes en une seule fois.

[43]:
```
if 'df' in locals():
    print("Distribution des valeurs brutes pour 'milieu_residence' :")
    display(df['milieu_residence'].value_counts(dropna=False))

    # Afficher les libellés correspondants
    variable_name_orig = 'mv025'
    if 'meta' in locals() and variable_name_orig in meta.variable_value_labels:
        print(f"\nLibellés pour '{variable_name_orig}':")
        for value, label in meta.variable_value_labels[variable_name_orig].
 ↪items():
            print(f"  {int(value)}: {label}")
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Distribution des valeurs brutes pour 'milieu_residence' :

milieu_residence
2.0    487
1.0    393
Name: count, dtype: int64
```

```
Libellés pour 'mv025':
  1: urban
  2: rural
```

Étape 1 (suite) : Interprétation

La variable est propre et ne contient que deux catégories : - `1.0` **(Urbain)** : 393 répondants. - `2.0` **(Rural)** : 487 répondants.

Il y a une légère prédominance du milieu rural dans notre échantillon du Borgou, ce qui est cohérent avec la démographie de la région.

Stratégie de Traitement : La variable est prête à l'emploi. Nous allons simplement la recoder avec des libellés clairs pour la visualisation.

Étape 2 : Nettoyage et Recodage

[44]:
```
if 'df' in locals():
    # Définir le mapping
    mapping_milieu = {
        1.0: 'Urbain',
        2.0: 'Rural'
    }

    # Créer la nouvelle variable
    df['milieu_residence_cat'] = df['milieu_residence'].map(mapping_milieu)

    print("Distribution de la variable recodée :")
    display(df['milieu_residence_cat'].value_counts())
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

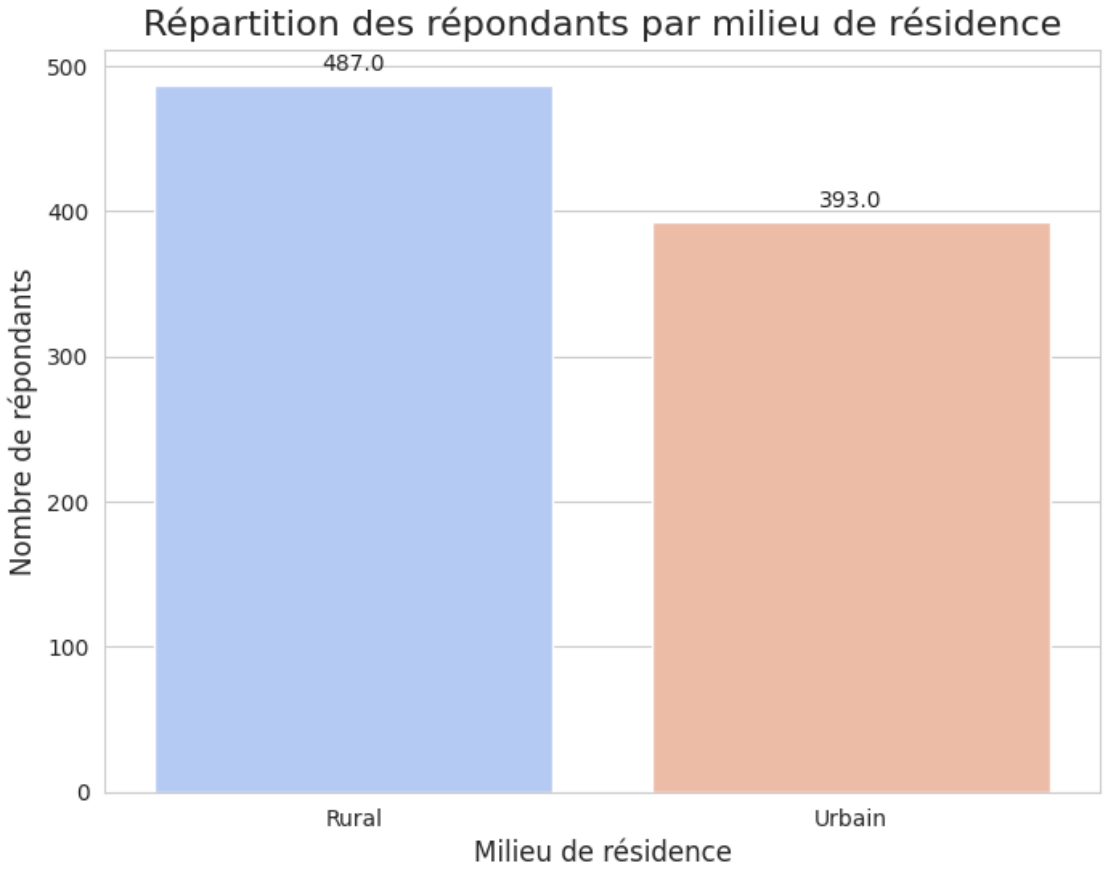
```
Distribution de la variable recodée :

milieu_residence_cat
Rural     487
Urbain    393
Name: count, dtype: int64
```

Étape 3 : Visualisation

[45]:
```
if 'df' in locals():
    plt.figure(figsize=(8, 6))
    ax = sns.countplot(x='milieu_residence_cat', data=df, palette='coolwarm')

    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                    ha='center', va='center', xytext=(0, 9), textcoords='offset␣
 ↪points')
```

```
    plt.title('Répartition des répondants par milieu de résidence', fontsize=16)
    plt.xlabel('Milieu de résidence', fontsize=12)
    plt.ylabel('Nombre de répondants', fontsize=12)
    plt.show()
else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
/tmp/ipykernel_53031/4130741902.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x='milieu_residence_cat', data=df, palette='coolwarm')
```



[46]:
```
if 'df' in locals():
    # Liste des variables de contrôle à inspecter
    variables_controle = [
```

```
        'statut_marital',
        'indice_richesse',
        'travaille_actuellement',
        'frequence_radio',
        'frequence_tv',
        'utilise_internet',
        'age_premier_rapport',
        'nb_partenaires_12mois'
    ]

    # Mapping vers les noms de variables originaux pour les libellés
    var_mapping = {
        'statut_marital': 'mv501',
        'indice_richesse': 'mv190',
        'travaille_actuellement': 'mv714',
        'frequence_radio': 'mv157',
        'frequence_tv': 'mv158',
        'utilise_internet': 'mv159a',
        'age_premier_rapport': 'mv525',
        'nb_partenaires_12mois': 'mv529'
    }

    for var in variables_controle:
        print(f"--- Variable : {var} ---")

        # Pour les variables numériques continues, on préfère describe()
        if var in ['age_premier_rapport', 'nb_partenaires_12mois']:
            display(df[var].describe())
        else:
            display(df[var].value_counts(dropna=False))

        # Afficher les libellés si disponibles
        original_var = var_mapping.get(var)
        if 'meta' in locals() and original_var and original_var in meta.
 ↪variable_value_labels:
            print(f"\nLibellés pour '{original_var}':")
            for value, label in meta.variable_value_labels[original_var].
 ↪items():
                print(f"  {int(value)}: {label}")

        print("\n" + "="*40 + "\n")

else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
--- Variable : statut_marital ---
```

```
statut_marital
```

```
1.0    546
0.0    301
2.0     17
4.0      7
5.0      7
3.0      2
Name: count, dtype: int64


Libellés pour 'mv501':
  0: never in union
  1: married
  2: living with partner
  3: widowed
  4: divorced
  5: no longer living together/separated

========================================

--- Variable : indice_richesse ---

indice_richesse
1.0    277
2.0    196
3.0    149
5.0    130
4.0    128
Name: count, dtype: int64


Libellés pour 'mv190':
  1: poorest
  2: poorer
  3: middle
  4: richer
  5: richest

========================================

--- Variable : travaille_actuellement ---

travaille_actuellement
1.0    817
0.0     63
Name: count, dtype: int64


Libellés pour 'mv714':
  0: no
  1: yes
```

```
========================================

--- Variable : frequence_radio ---

frequence_radio
2.0    445
0.0    259
1.0    176
Name: count, dtype: int64


Libellés pour 'mv157':
  0: not at all
  1: less than once a week
  2: at least once a week
  3: almost every day

========================================

--- Variable : frequence_tv ---

frequence_tv
0.0    453
2.0    246
1.0    181
Name: count, dtype: int64


Libellés pour 'mv158':
  0: not at all
  1: less than once a week
  2: at least once a week
  3: almost every day

========================================

--- Variable : utilise_internet ---

utilise_internet
0.0    769
1.0    104
2.0      7
Name: count, dtype: int64


========================================

--- Variable : age_premier_rapport ---

count    880.000000
```

```
mean      15.255682
std        7.196216
min        0.000000
25%       15.000000
50%       18.000000
75%       20.000000
max       30.000000
Name: age_premier_rapport, dtype: float64


Libellés pour 'mv525':
  0: not had sex
  96: at first union
  97: inconsistent
  98: don't know

========================================

--- Variable : nb_partenaires_12mois ---

count    880.000000
mean       0.952273
std        0.737424
min        0.000000
25%        1.000000
50%        1.000000
75%        1.000000
max        8.000000
Name: nb_partenaires_12mois, dtype: float64


Libellés pour 'mv529':
  995: within last 4 weeks
  996: before last birth
  997: inconsistent
  998: don't know

========================================
```

2.3.3 5.3.3. Recodage Groupé des Variables de Contrôle

En nous basant sur l'inspection précédente, nous allons maintenant recoder toutes les variables de contrôle en une seule étape pour préparer le jeu de données à l'analyse bivariée et à la modélisation.

[47]:
```python
if 'df' in locals():
    # 1. statut_marital
    mapping_marital = {
```

```
        0.0: 'Jamais en couple',
        1.0: 'En couple',
        2.0: 'En couple',
        3.0: 'A été en couple',
        4.0: 'A été en couple',
        5.0: 'A été en couple'
    }
    df['statut_marital_cat'] = df['statut_marital'].map(mapping_marital)

    # 2. indice_richesse
    mapping_richesse = {
        1.0: 'Plus pauvre',
        2.0: 'Pauvre',
        3.0: 'Moyen',
        4.0: 'Riche',
        5.0: 'Plus riche'
    }
    df['indice_richesse_cat'] = df['indice_richesse'].map(mapping_richesse)

    # 3. travaille_actuellement
    df['travaille_actuellement_cat'] = df['travaille_actuellement'].map({1.0:␣
↪'Oui', 0.0: 'Non'})

    # 4. frequence_radio & 5. frequence_tv
    mapping_media = {
        0.0: 'Pas du tout',
        1.0: 'Moins d\'une fois/semaine',
        2.0: 'Au moins une fois/semaine',
        3.0: 'Au moins une fois/semaine'
    }
    df['frequence_radio_cat'] = df['frequence_radio'].map(mapping_media)
    df['frequence_tv_cat'] = df['frequence_tv'].map(mapping_media)

    # 6. utilise_internet
    df['utilise_internet_cat'] = df['utilise_internet'].map({1.0: 'Oui', 2.0:␣
↪'Oui', 0.0: 'Non'})

    # 7. age_premier_rapport -> activité sexuelle
    df['sexuellement_actif'] = np.where(df['age_premier_rapport'] > 0, 'Oui',␣
↪'Non')

    # 8. nb_partenaires_12mois
    def categorize_partners(n):
        if n == 0:
            return '0'
        elif n == 1:
            return '1'
```

```
        else:
            return '2 ou plus'
    df['nb_partenaires_12mois_cat'] = df['nb_partenaires_12mois'].
↪apply(categorize_partners)

    print("Les variables de contrôle ont été recodées avec succès.")

    # Afficher un aperçu des nouvelles colonnes
    new_cols = [
        'statut_marital_cat', 'indice_richesse_cat',␣
↪'travaille_actuellement_cat',
        'frequence_radio_cat', 'frequence_tv_cat', 'utilise_internet_cat',
        'sexuellement_actif', 'nb_partenaires_12mois_cat'
    ]
    display(df[new_cols].head())

else:
    print("Le DataFrame 'df' n'est pas chargé.")
```

```
Les variables de contrôle ont été recodées avec succès.
```

	statut_marital_cat	indice_richesse_cat	travaille_actuellement_cat	\
0	En couple	Pauvre	Oui	
1	En couple	Pauvre	Oui	
2	En couple	Pauvre	Oui	
3	Jamais en couple	Plus pauvre	Oui	
4	En couple	Plus pauvre	Oui	

	frequence_radio_cat	frequence_tv_cat	utilise_internet_cat	\
0	Pas du tout	Pas du tout	Non	
1	Au moins une fois/semaine	Moins d'une fois/semaine	Non	
2	Au moins une fois/semaine	Pas du tout	Non	
3	Au moins une fois/semaine	Moins d'une fois/semaine	Non	
4	Pas du tout	Pas du tout	Non	

	sexuellement_actif	nb_partenaires_12mois_cat
0	Oui	1
1	Oui	1
2	Oui	2 ou plus
3	Oui	0
4	Oui	1