mooc_mertes

July 1, 2023

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import forestplot as fp
import missingno as msno
from statsmodels.graphics.mosaicplot import mosaic
from statsmodels.formula.api import ols
from statsmodels.formula.api import logit
from statsmodels.formula.api import glm
from statsmodels.api import qqplot
from statsmodels.stats.outliers_influence import OLSInfluence
from scipy.stats import ttest_ind as t_student
from scipy.stats import mannwhitneyu
from scipy.stats import chi2_contingency
from scipy.stats import pearsonr
from scipy.stats import spearmanr
from scipy.stats import kstest
from scipy.stats import norm
from scipy.stats import zscore
from matplotlib.patches import Rectangle
```

[ ]:
```
# Chargement des données
effec1 = pd.read_csv("../csv/effec1.quest.compil.csv", encoding="ISO-8859-1")
effec2 = pd.read_csv("../csv/effec2.quest.compil.csv", encoding="ISO-8859-1")
effec3 = pd.read_csv("../csv/effec3.quest.compil.csv", encoding="ISO-8859-1")
#
usage1 = pd.read_csv("../csv/usages.effec1.csv", encoding="ISO-8859-1")
usage2 = pd.read_csv("../csv/usages.effec2.csv", encoding="ISO-8859-1")
usage3 = pd.read_csv("../csv/usages.effec3.csv", encoding="ISO-8859-1")
```

[ ]:
```
usage1.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7965 entries, 0 to 7964
Data columns (total 73 columns):
 #   Column            Non-Null Count  Dtype
```

```
---  ------            --------------  -----
 0   Student_ID        7965 non-null   int64
 1   Exam.score        20 non-null     float64
 2   Exam.bin          7965 non-null   int64
 3   Assignment.score  2455 non-null   float64
 4   Assignment.bin    7965 non-null   int64
 5   Quizz.1.score     4649 non-null   float64
 6   Quizz.1.bin       7965 non-null   int64
 7   Quizz.2.score     3762 non-null   float64
 8   Quizz.2.bin       7965 non-null   int64
 9   Quizz.3.score     3304 non-null   float64
 10  Quizz.3.bin       7965 non-null   int64
 11  Quizz.4.bin       7965 non-null   int64
 12  Quizz.4.score     2971 non-null   float64
 13  Quizz.5.bin       7965 non-null   int64
 14  Quizz.5.score     2853 non-null   float64
 15  Intro.MOOC        0 non-null      float64
 16  Prez.sem.1        7965 non-null   int64
 17  S1.L1             7965 non-null   int64
 18  S1.L2             7965 non-null   int64
 19  S1.L3             7965 non-null   int64
 20  S1.L4             7965 non-null   int64
 21  S1.L5             7965 non-null   int64
 22  S1.L6             7965 non-null   int64
 23  Prez.sem.2        7965 non-null   int64
 24  S2.L1             7965 non-null   int64
 25  S2.L2             7965 non-null   int64
 26  S2.L3             7965 non-null   int64
 27  S2.L4             7965 non-null   int64
 28  S2.L5             7965 non-null   int64
 29  S2.L6             7965 non-null   int64
 30  Prez.sem.3        7965 non-null   int64
 31  S3.L1.1           7965 non-null   int64
 32  S3.L1.2           7965 non-null   int64
 33  S3.L2             7965 non-null   int64
 34  S3.L3             7965 non-null   int64
 35  S3.L4             7965 non-null   int64
 36  S3.L5             7965 non-null   int64
 37  Prez.sem.4        7965 non-null   int64
 38  S4.L1.1           7965 non-null   int64
 39  S4.L1.2           7965 non-null   int64
 40  S4.L2             7965 non-null   int64
 41  S4.L3             7965 non-null   int64
 42  S4.L4             7965 non-null   int64
 43  S4.L5             7965 non-null   int64
 44  Prez.sem.5        7965 non-null   int64
 45  S5.L1.1           7965 non-null   int64
 46  S5.L1.2           7965 non-null   int64
```

```
 47  S5.L2             7965 non-null   int64
 48  S5.L3             7965 non-null   int64
 49  S5.L4             7965 non-null   int64
 50  S5.L5             7965 non-null   int64
 51  Post.forum.0      7965 non-null   int64
 52  view.forum.0      7965 non-null   int64
 53  Post.forum.1      7965 non-null   int64
 54  Post.forum.1.2    7965 non-null   int64
 55  view.forum.1      7965 non-null   int64
 56  view.forum.1.2    7965 non-null   int64
 57  Post.forum.2      7965 non-null   int64
 58  Post.forum.2.2    7965 non-null   int64
 59  view.forum.2      7965 non-null   int64
 60  view.forum.2.2    7965 non-null   int64
 61  Post.forum.3      7965 non-null   int64
 62  view.forum.3      7965 non-null   int64
 63  Post.forum.4      7965 non-null   int64
 64  Post.forum.4.2    7965 non-null   int64
 65  view.forum.4      7965 non-null   int64
 66  view.forum.4.2    7965 non-null   int64
 67  Post.forum.5      7965 non-null   int64
 68  Post.forum.5.2    7965 non-null   int64
 69  view.forum.5      7965 non-null   int64
 70  view.forum.5.2    7965 non-null   int64
 71  last.video        7965 non-null   int64
 72  last.quizz        7965 non-null   int64
dtypes: float64(8), int64(65)
memory usage: 4.4 MB
```

[ ]:
```
effec1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8986 entries, 0 to 8985
Data columns (total 35 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Student_ID        8986 non-null   int64
 1   Gender            5342 non-null   object
 2   birth.year        5107 non-null   float64
 3   Country           5303 non-null   object
 4   Diploma           5328 non-null   object
 5   Formation         5306 non-null   object
 6   CSP               5320 non-null   object
 7   How.heard         5326 non-null   object
 8   Exp.crea          5336 non-null   object
 9   Curiosity.MOOC    5310 non-null   float64
 10  Certif.self.sat   5290 non-null   float64
 11  Rencontres        5287 non-null   float64
```

```
 12  Certif.work        5265 non-null   float64
 13  Incitation         5268 non-null   float64
 14  Temps.Dispo        5315 non-null   object
 15  Exp.MOOC           5302 non-null   object
 16  Completion.proba   5324 non-null   float64
 17  Instit.brand       2648 non-null   object
 18  motiv.princ        2649 non-null   object
 19  diffic             2648 non-null   object
 20  encad.disp         2635 non-null   object
 21  How.contact        2644 non-null   object
 22  entour             2643 non-null   object
 23  entour.inter       1874 non-null   object
 24  Satisf             2645 non-null   float64
 25  Eval.diffic        2646 non-null   object
 26  Estimated.hours    2645 non-null   object
 27  Part.labo          2639 non-null   object
 28  Plat.satisf        2639 non-null   object
 29  Peer.eval.relev    2637 non-null   object
 30  encad.diffic       0 non-null      float64
 31  Country_HDI        5247 non-null   object
 32  Country_HDI.fin    5247 non-null   object
 33  CSP.fin            5320 non-null   object
 34  Temps.dispo.fin    5315 non-null   object
dtypes: float64(9), int64(1), object(25)
memory usage: 2.4+ MB
```

[ ]:
```
# Fusion des tables effec[n] et usage[n] dans des bases intermédiaires selon les identifiants des étudiants
bases_inter = []
for tabs in [(effec1, usage1), (effec2, usage2), (effec3, usage3)]:
    bases_inter.append(tabs[0].merge(tabs[1], on="Student_ID"))
```

[ ]:
```
# Concaténation des tables intermédiaires afin de créer une seule base commune
base = pd.concat(bases_inter, join="outer",
                 axis=0, ignore_index=False,
                 keys=[1, 2, 3])
```

[ ]:
```
base = base.reset_index(level=0).rename({"level_0": "Itération"}, axis=1)
```

[ ]:
```
# Indication de non passage de la certification pour les itération 1 et 2 où seul l'examen existe.
base[["Exam.bin", "Certif.bin"]] = base[["Exam.bin", "Certif.bin"]].fillna(0)
```

[ ]:
```
base[["Certif.bin", "Exam.bin"]]
```

[ ]:
```
      Certif.bin  Exam.bin
0            0.0         0
1            0.0         0
```

```
2             0.0         0
3             0.0         0
4             0.0         0
…             …         …
3510          0.0         0
3511          0.0         0
3512          0.0         0
3513          0.0         0
3514          0.0         0

[15182 rows x 2 columns]
```

[ ]:
```
base = base.astype({"Exam.bin": bool, "Certif.bin": bool, "Student_ID": int})
```

[ ]:
```
base = base.copy()
# Map des modalités M et H de la variable Country_HDI
base["New_HDI"] = np.select([base["Country_HDI"] == "M",
                             base["Country_HDI"] == "H"],
                            ["I", "I"], default=base["Country_HDI"])
```

[ ]:
```
# Sélection des vidéos
df_video = base.set_index(["Itération", "Student_ID",
                           "Gender", "New_HDI"]).loc[:, 'S1.L1':'S5.L5'].
↪drop(["Prez.sem.2", "Prez.sem.3",
                                                                        ␣
↪"Prez.sem.4", "Prez.sem.5"], axis=1)
```

[ ]:
```
# Sélection des Quiz
df_quiz = base.set_index(["Itération", "Student_ID",
                           "Gender", "New_HDI"]).loc[:, 'Quizz.1.bin':'Quizz.5.
↪bin'].drop(["Quizz.2.score",
                                                                        ␣
↪            "Quizz.3.score",
                                                                        ␣
↪            "Quizz.4.score"], axis=1)
```

[ ]:
```
# Sélection des passages des Examens, certifications et assignment
df_exam = base.set_index(["Itération", "Student_ID",
                           "Gender", "New_HDI"])[["Exam.bin", "Certif.bin",␣
↪"Assignment.bin"]]
```

[ ]:
```
df_exam
```

[ ]:
```
                                      Exam.bin  Certif.bin  Assignment.bin
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN        False       False               0
          19178      une femme TH         False       False               0
```

```
            1086       une femme TH          False       False              0
            1948       une femme TH          False       False              0
            16209      une femme B           False       False              0
…                                             …           …                  …
3           42092      un homme  B           False       False              1
            64673      NaN       NaN         False       False              0
            67894      NaN       NaN         False       False              0
            66874      NaN       NaN         False       False              0
            66492      NaN       NaN         False       False              0

[15182 rows x 3 columns]
```

[ ]:
```
# Nombre de videos par apprenant pour l'ensemble du MOOC.
total_video = pd.DataFrame(df_video.sum(axis=1), columns=["video"])

# Nombre de questionnaires par apprenant pour l'ensemble du MOOC.
total_quiz = pd.DataFrame(df_quiz.sum(axis=1), columns=["quiz"])
```

[ ]:
```
total_video.head()
```

[ ]:
```
                                        video
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN         0
          19178      une femme TH          1
          1086       une femme TH         30
          1948       une femme TH          1
          16209      une femme B           0
```

[ ]:
```
total_quiz.head()
```

[ ]:
```
                                        quiz
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN         0
          19178      une femme TH          0
          1086       une femme TH          4
          1948       une femme TH          0
          16209      une femme B           5
```

[ ]:
```
# Création de la table regroupant toutes la variables pour mesurer l'engagement␣
 ↪de chaque apprenant
total_student = pd.concat([total_video, total_quiz, df_exam], axis=1)
```

[ ]:
```
total_student.head()
```

[ ]:
```
                                        video  quiz  Exam.bin  Certif.bin  \
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN         0     0     False       False
```

```
           19178      une femme TH           1     0     False       False
           1086       une femme TH          30     4     False       False
           1948       une femme TH           1     0     False       False
           16209      une femme B            0     5     False       False

                                          Assignment.bin
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN                   0
          19178      une femme TH                    0
          1086       une femme TH                    0
          1948       une femme TH                    0
          16209      une femme B                     0
```

[ ]:
```python
# selection des types d'apprenant
def student_type(col):
    video, quiz, exam, certif, devoir = col
    if (exam >= 1 or certif >= 1):
        return "Completer"
    elif quiz > 0 and devoir > 0:
        return "Disengaging"
    elif video > 6:
        return "Auditing"
    else:
        return "Bystander"
```

[ ]:
```python
total_student["Type"] = total_student.apply(student_type, axis=1)
```

[ ]:
```python
total_student.head()
```

[ ]:
```
                                          video  quiz  Exam.bin  Certif.bin  \
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN           0     0     False       False
          19178      une femme TH            1     0     False       False
          1086       une femme TH           30     4     False       False
          1948       une femme TH            1     0     False       False
          16209      une femme B             0     5     False       False

                                          Assignment.bin       Type
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN                   0  Bystander
          19178      une femme TH                    0  Bystander
          1086       une femme TH                    0   Auditing
          1948       une femme TH                    0  Bystander
          16209      une femme B                     0  Bystander
```

[ ]:
```python
student = total_student.reset_index()[["Student_ID", "Type", "Itération"]]
```

[ ]:
```
# Calcul du nombre d'apprenants par type et par itération
df_type = student.groupby(["Itération", "Type"])[["Type"]].count().
↪rename({'Type': 'total'}, axis=1)
```

[ ]:
```
df_type
```

[ ]:
```
                      total
Itération Type
1         Auditing      1207
          Bystander     4285
          Completer       20
          Disengaging   2453
2         Auditing       538
          Bystander     2168
          Completer      876
          Disengaging    120
3         Auditing       375
          Bystander     2238
          Completer      832
          Disengaging     70
```

[ ]:
```
df_type.reset_index("Type", inplace=True)
```

[ ]:
```
# Nombre total d'apprenants par itération
df_iter = df_type.groupby("Itération").sum()
```

[ ]:
```
df_iter
```

[ ]:
```
           total
Itération
1           7965
2           3702
3           3515
```

[ ]:
```
total_iter = df_type.merge(df_iter, on="Itération", suffixes=["_type", "_iter"])
```

[ ]:
```
total_iter
```

[ ]:
```
                  Type  total_type  total_iter
Itération
1             Auditing        1207        7965
1            Bystander        4285        7965
1            Completer          20        7965
1          Disengaging        2453        7965
2             Auditing         538        3702
2            Bystander        2168        3702
2            Completer         876        3702
2          Disengaging         120        3702
```

```
3             Auditing         375        3515
3            Bystander        2238        3515
3            Completer         832        3515
3          Disengaging          70        3515
```

[ ]:
```
# Proportion d'apprenants par types d'apprenants et par itération
total_iter["proportion/iter"] = round(total_iter["total_type"] /␣
 ↪total_iter["total_iter"] * 100, 1)
```

[ ]:
```
total_iter
```

[ ]:
```
                  Type  total_type  total_iter  proportion/iter
Itération
1             Auditing        1207        7965             15.2
1            Bystander        4285        7965             53.8
1            Completer          20        7965              0.3
1          Disengaging        2453        7965             30.8
2             Auditing         538        3702             14.5
2            Bystander        2168        3702             58.6
2            Completer         876        3702             23.7
2          Disengaging         120        3702              3.2
3             Auditing         375        3515             10.7
3            Bystander        2238        3515             63.7
3            Completer         832        3515             23.7
3          Disengaging          70        3515              2.0
```

[ ]:
```
base["Genre"] = base["Gender"].map({"un homme": "Homme", "une femme": "Femme"})
```

[ ]:
```
# Tableau de contingence (croisement des 2 variables catégorielles)
tab_obs = pd.crosstab(index=base["Genre"], columns=base["New_HDI"])
```

[ ]:
```
tab_obs.rename(columns={"un homme": "H", "une femme": "F"}, inplace=True)
```

[ ]:
```
tab_obs
```

[ ]:
```
New_HDI    B    I    TH
Genre
Femme    147  233  2545
Homme    883  432  4711
```

[ ]:
```
# Test d'indépendance (chi2)
chi2, p_value, degres_liberte, tableau_attendu = chi2_contingency(tab_obs)
```

[ ]:
```
tableau_attendu
```

[ ]:
```
array([[ 336.58250475,  217.308122  , 2371.10937325],
       [ 693.41749525,  447.691878  , 4884.89062675]])
```

[ ]:
```
chi2
```

[ ]: 179.2420322171424

[ ]:
```
p_value
```

[ ]: 1.196980957821505e-39

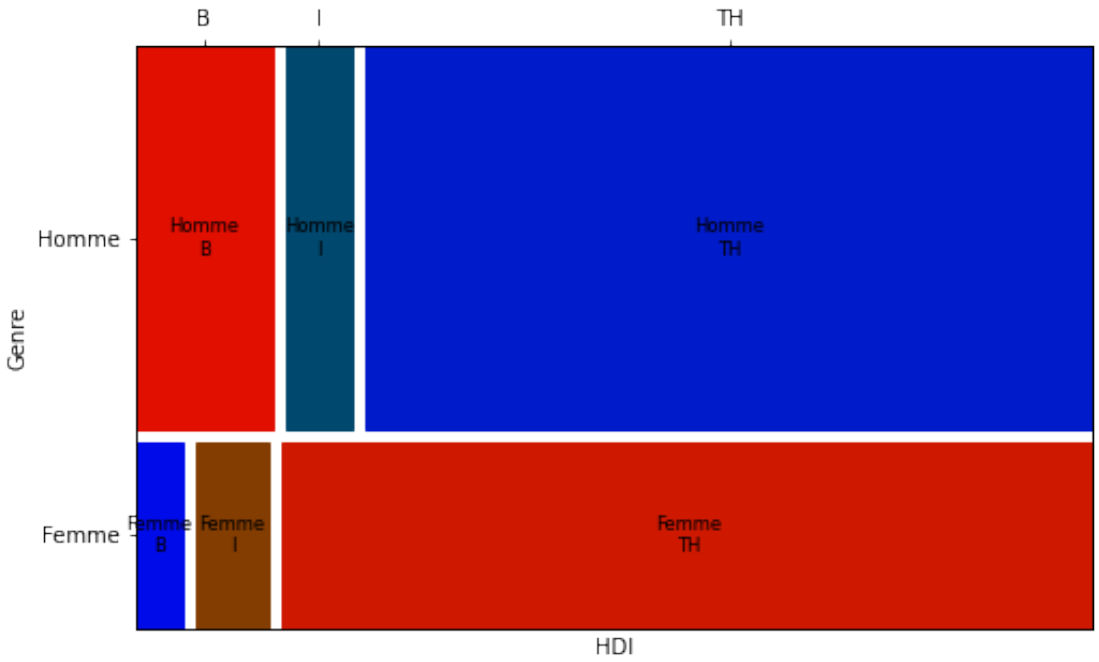
La p value est $< 5\%$ indiquerait que l'index HDI serait significativement lié au genre puisqu'il y a moins de 5 % de chance que les 2 variables soient indépendantes.

[ ]:
```
#residus = (tab_HDI_gender - tableau_attendu) / tableau_attendu
residus = tab_obs - tableau_attendu
```

[ ]:
```
residus
```

[ ]:
```
New_HDI           B          I          TH
Genre
Femme    -189.582505  15.691878  173.890627
Homme     189.582505 -15.691878 -173.890627
```

[ ]:
```
# Mosaic du tableau de contingence
#props = lambda key: {'color': 'red' if residus(key[0]) > 0 else 'blue'},
fig, ax = plt.subplots(figsize=(8,5))
mosaic(tab_obs.stack(), statistic=True, gap=0.02, horizontal=False, ax=ax)
ax.set_xlabel("HDI")
ax.set_ylabel("Genre")
plt.savefig("../graph/mosaic_contingence.png")
plt.show()
```



[ ]:
```
# valeurs attendues sous l'hypothèse nulle (H0).
x = tableau_attendu.flatten()
```

[ ]:
```
x
```

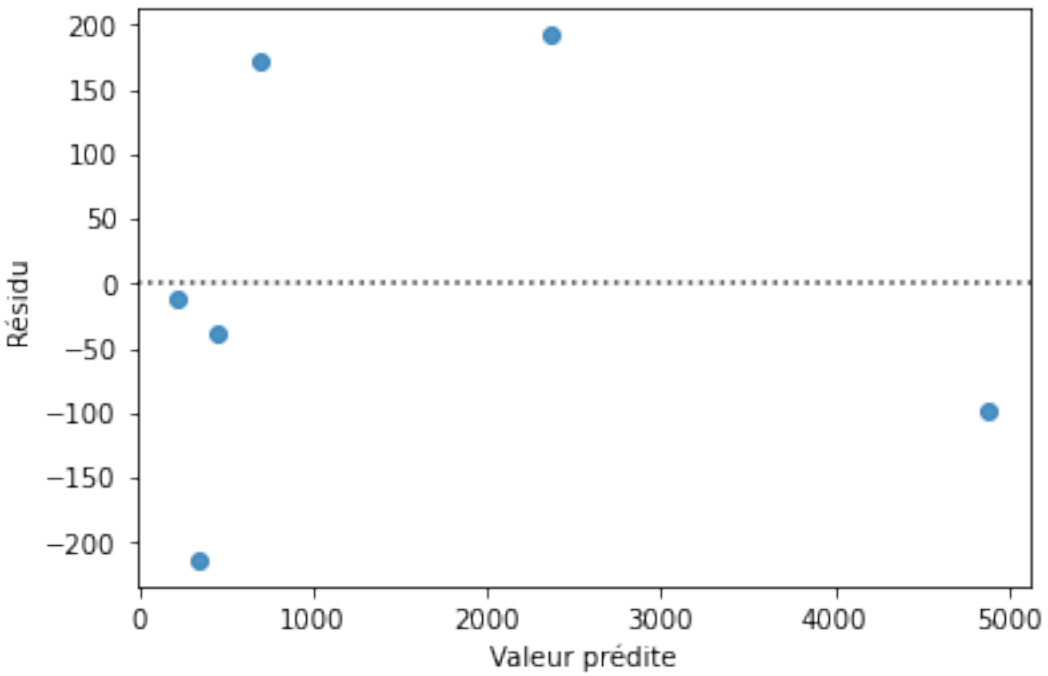
[ ]:
```
array([ 336.58250475,  217.308122  , 2371.10937325,  693.41749525,
        447.691878  , 4884.89062675])
```

[ ]:
```
# valeurs observées
y = tab_obs.stack().values
```

[ ]:
```
y
```

[ ]:
```
array([ 147,  233, 2545,  883,  432, 4711])
```

[ ]:
```
# Résidus du modèle de prédiction
fig, ax = plt.subplots()
sns.residplot(x=x, y=y, ax=ax)
#ax.set_title("Résidus du modèle observé")
ax.set_ylabel("Résidu")
ax.set_xlabel("Valeur prédite")
plt.savefig("../graph/residus_chi2.png")
plt.show()
```



fig, ax = plt.subplots() sns.heatmap(residus, annot=True, cmap='coolwarm', cbar=True, ax=ax) ax.set_ylabel("Genre") ax.set_ylabel("HDI") plt.show()

formule du V de Cramer :

V = √(² / (n * (min(r, c) - 1)))

Dans cette formule :

V représente le coefficient de Cramer. ² est la statistique du chi carré. n est la taille de l'échantillon. r est le nombre de niveaux ou de catégories de la première variable. c est le nombre de niveaux ou de catégories de la deuxième variable.

[ ]:
```
# Fonction de calcul du V de Cramer
# data = tab ndarray
def V_Cramer(data):
    # somme de chaque colonne
    n = np.sum(data)
    # taille du tableau de contingence des variables catégorielles (taille de␣
 ↪chaque échantillon pour chaque variable)
    row, col = tab_obs.shape
    # Formule du V de Cramer
    V = np.sqrt(chi2 / (n * (min([row, col]) - 1)))
    return V
```

[ ]:
```
V_Cramer(np.array(tab_obs))
```

[ ]:
```
0.14150902903141144
```

[ ]:
```
V_Cramer(tableau_attendu)
```

[ ]:
```
0.14150902903141144
```

La valeur V de Cramer étant faible il y a aurait une faible dépendance entre l'index HDI et le genre. Il y aurait donc statistiquement une association entre ses deux variables catégorielles, indiquée par la p-value du chi2, mais la valeur du V de Cramer indiquerait que cette dépendance serait faible.

[ ]:
```
# tableau du genre par étudiant
genre_etu = base[["Student_ID", "Genre"]].drop_duplicates(subset="Student_ID").
 ↪dropna()
```

[ ]:
```
genre_etu.head()
```

[ ]:
```
   Student_ID  Genre
1       19178  Femme
2        1086  Femme
3        1948  Femme
4       16209  Femme
```

```
5        6685  Homme
```

[ ]:
```
# Nombre total de videos par étudiant
total_video_etu = total_video.groupby("Student_ID").sum()
```

[ ]:
```
total_video_etu.head()
```

[ ]:
```
            video
Student_ID
15              3
28              0
34              0
36              0
45             25
```

[ ]:
```
# tableau du genre et du nombre total de videos visionnées par étudiant
etu_genre_video = total_video_etu.merge(genre_etu, on='Student_ID')
```

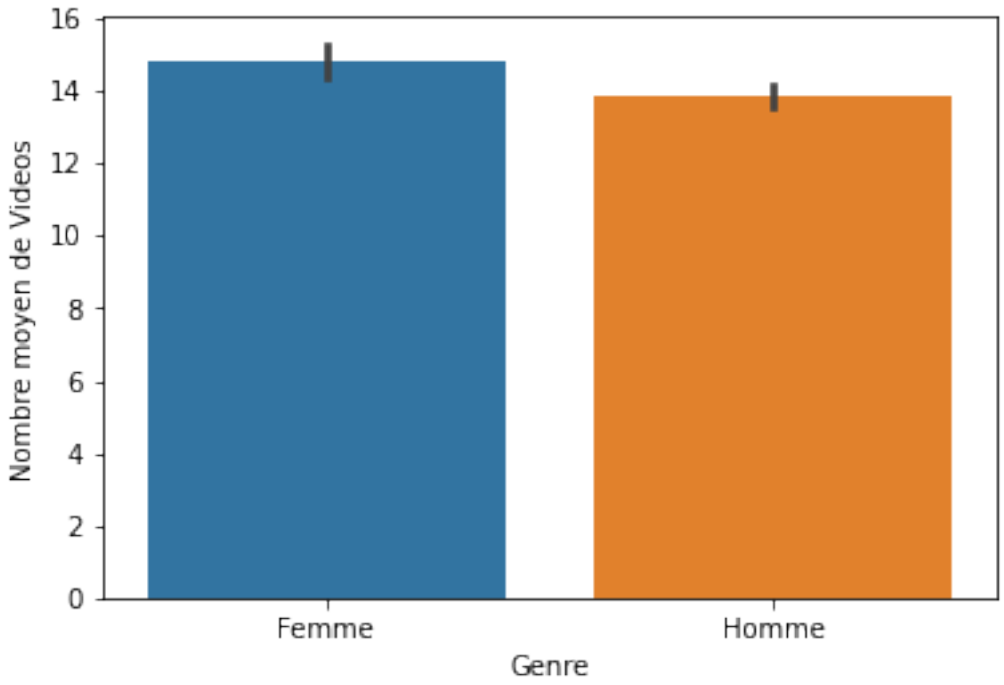
[ ]:
```
etu_genre_video.head()
```

[ ]:
```
   Student_ID  video  Genre
0          45     25  Femme
1          83     22  Homme
2          84      8  Homme
3          87      1  Homme
4          94      2  Homme
```

[ ]:
```
# moyenne du nombre de videos visionnées par genre
total_video_etu.merge(genre_etu, on="Student_ID").groupby("Genre")[["video"]].
↪mean()
```

[ ]:
```
            video
Genre
Femme  14.855376
Homme  13.867276
```

[ ]:
```
# Figure de la distribution du nombre de video par rapport au genre
fig, ax = plt.subplots()
sns.barplot(data=etu_genre_video, x="Genre", y="video", ax=ax)
ax.set_xlabel("Genre")
ax.set_ylabel("Nombre moyen de Videos")
plt.savefig("../graph/mean_video.png")
plt.show()
```



H0 (Hypothèse nulle) : il n'y a pas de différence sur le nombre de video visionnées entre les hommes et les femmes

[ ]:
```
tab_stat = etu_genre_video.pivot_table(columns="Genre", index="Student_ID",␣
↪values="video").fillna(0)
```

[ ]:
```
tab_stat.head()
```

[ ]:
```
Genre       Femme  Homme
Student_ID
45           25.0    0.0
83            0.0   22.0
84            0.0    8.0
87            0.0    1.0
94            0.0    2.0
```

[ ]:
```
# Test de Student
statistique, p_value = t_student(tab_stat["Homme"], tab_stat["Femme"])
```

[ ]:
```
statistique
```

[ ]:
```
26.98492395204523
```

[ ]:
```
p_value
```

[ ]:
```
3.4121378553908065e-157
```

Il y a moins de 5% de chance qu'il n'y ait pas de différence du nombre de visionnages entre les femmes et les hommes. Il y aurait significativement un lien entre le nombre de visionnages et le genre.

[ ]:
```
# Sélection par genre
df_hom = etu_genre_video[etu_genre_video["Genre"] == "Homme"]
df_fem = etu_genre_video[etu_genre_video["Genre"] == "Femme"]
```

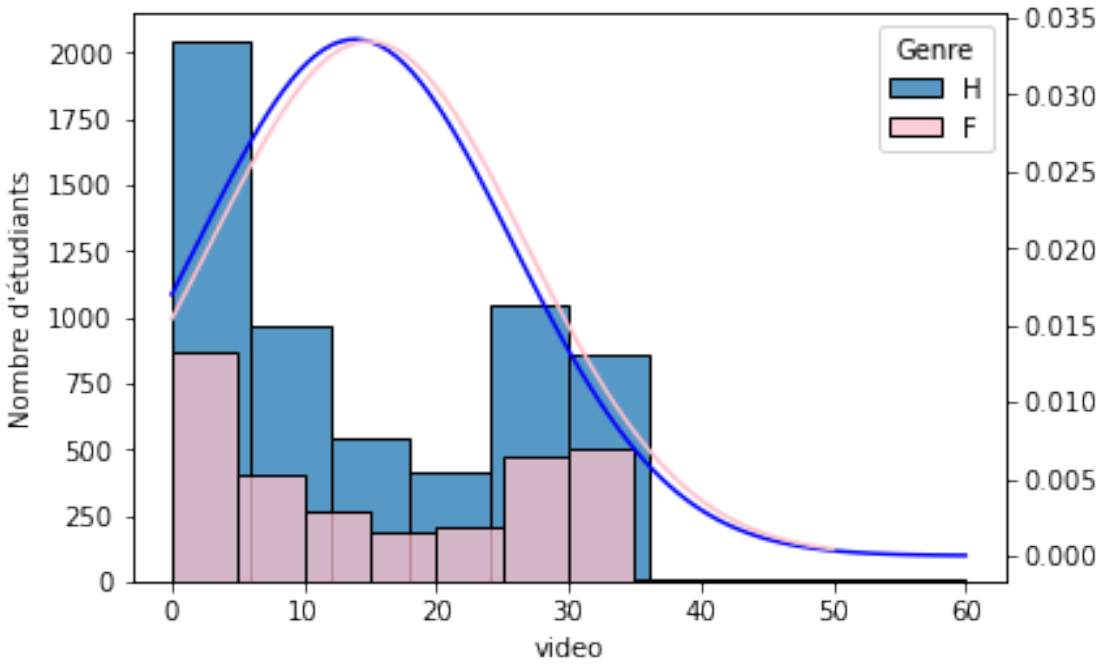
[ ]:
```
df_hom
```

[ ]:
```
      Student_ID  video  Genre
1             83     22  Homme
2             84      8  Homme
3             87      1  Homme
4             94      2  Homme
5             98     23  Homme
…            …     …     …
8807       68205     30  Homme
8808       68220     30  Homme
8809       68223      0  Homme
8811       68265     14  Homme
8813       68282      1  Homme

[5907 rows x 3 columns]
```

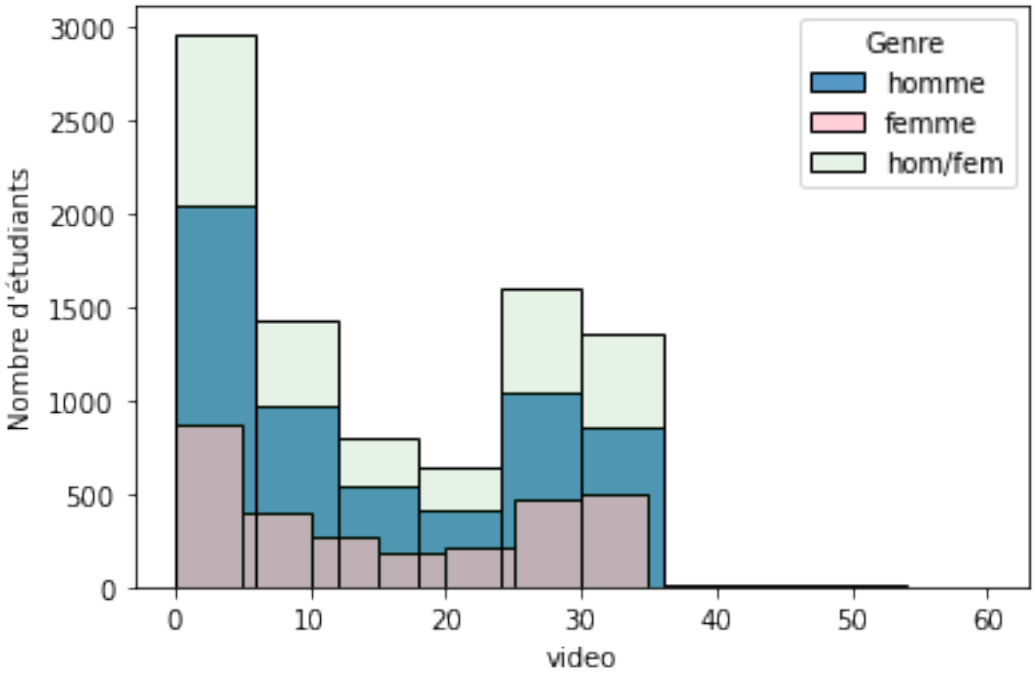
[ ]:
```
# Calcul des paramètres de la distribution normale pour chaque groupe
mu_hommes, std_hommes = np.mean(df_hom["video"]), np.std(df_hom["video"])
mu_femmes, std_femmes = np.mean(df_fem["video"]), np.std(df_fem["video"])

# Création des valeurs x pour tracer la courbe théorique
x_hommes = np.linspace(min(df_hom["video"]), max(df_hom["video"]), 100)
x_femmes = np.linspace(min(df_fem["video"]), max(df_fem["video"]), 100)

# Calcul des valeurs y correspondantes en utilisant la PDF de la distribution␣
 ↪normale
y_hommes = norm.pdf(x_hommes, mu_hommes, std_hommes)
y_femmes = norm.pdf(x_femmes, mu_femmes, std_femmes)
```

[ ]:
```
figure, ax1 = plt.subplots()
sns.histplot(data=df_hom, x="video", bins=10, color="tab:blue", ax=ax1,␣
 ↪label="H")
sns.histplot(data=df_fem, x="video", bins=10, color="pink", ax=ax1, label="F")
ax1.set_xlabel("video")
ax1.set_ylabel("Nombre d'étudiants")
ax1.legend(title="Genre")
ax2 = ax1.twinx()
```

```
ax2.plot(x_hommes, y_hommes, color='blue')
ax2.plot(x_femmes, y_femmes, color='pink')
plt.savefig("../graph/distribution_video.png")
plt.show()
```



[ ]:
```
# visualisation de la normalité de la distribution des données (histogramme de distribution)
figure, ax = plt.subplots()
sns.histplot(data=df_hom, x="video", bins=10, color="tab:blue", ax=ax, label="homme")
sns.histplot(data=df_fem, x="video", bins=10, color="pink", ax=ax, label="femme")
sns.histplot(data=etu_genre_video, x="video", bins=10, color="green", ax=ax, label="hom/fem", alpha=0.1)
ax.set_xlabel("video")
ax.set_ylabel("Nombre d'étudiants")
ax.legend(title="Genre")
plt.savefig("../graph/distribution_video2.png")
plt.show()
```



La distribution des données ne suit pas une loi normale. Ce qui ne permet pas de faire un test t-Student puisque la condition première est que les données doivent être normalement distribuées.

[ ]:
```
# Models de regression du nombre de video selon le genre
mdl_video_vs_genre = ols("video ~ Genre", data=etu_genre_video).fit()
```

[ ]:
```
mdl_video_vs_genre.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  video   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     13.45
Date:                Sat, 01 Jul 2023   Prob (F-statistic):           0.000247
Time:                        08:35:48   Log-Likelihood:                -34348.
No. Observations:                8818   AIC:                         6.870e+04
Df Residuals:                    8816   BIC:                         6.871e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025
0.975]
```

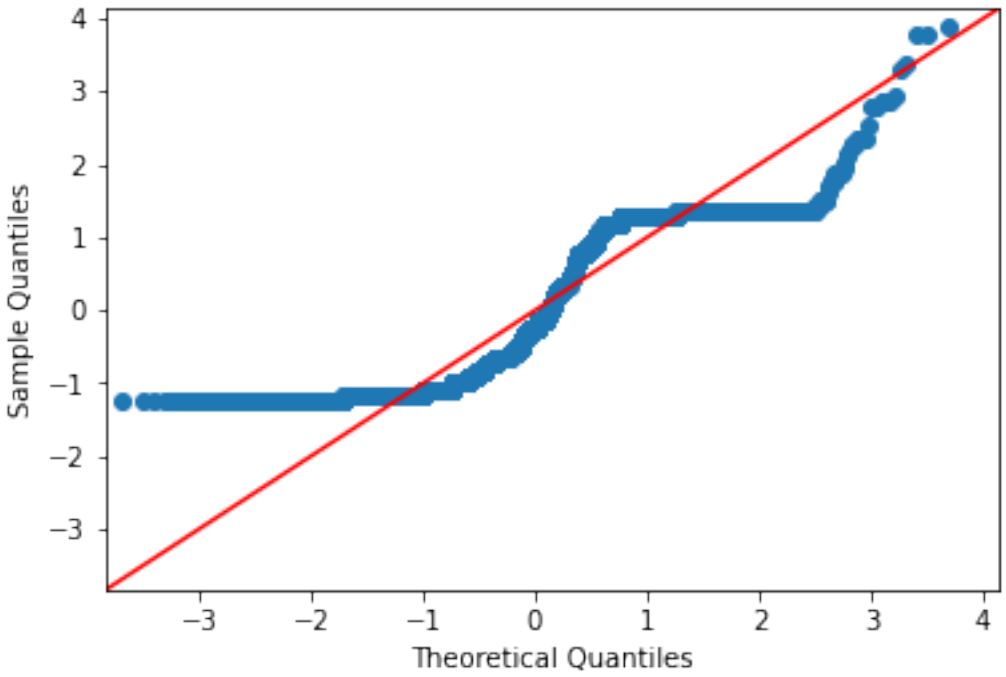
```
------------------------------------------------------------------------------
--
Intercept          14.8554      0.221     67.365      0.000      14.423
15.288
Genre[T.Homme]     -0.9881      0.269     -3.667      0.000      -1.516
-0.460
==============================================================================
Omnibus:                    59307.238   Durbin-Watson:                   1.962
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              878.417
Skew:                           0.240   Prob(JB):                    1.80e-191
Kurtosis:                       1.530   Cond. No.                         3.24
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Le coefficient R^2 étant proche de 0, le model de régression est peu fidèle à l'observation des données.

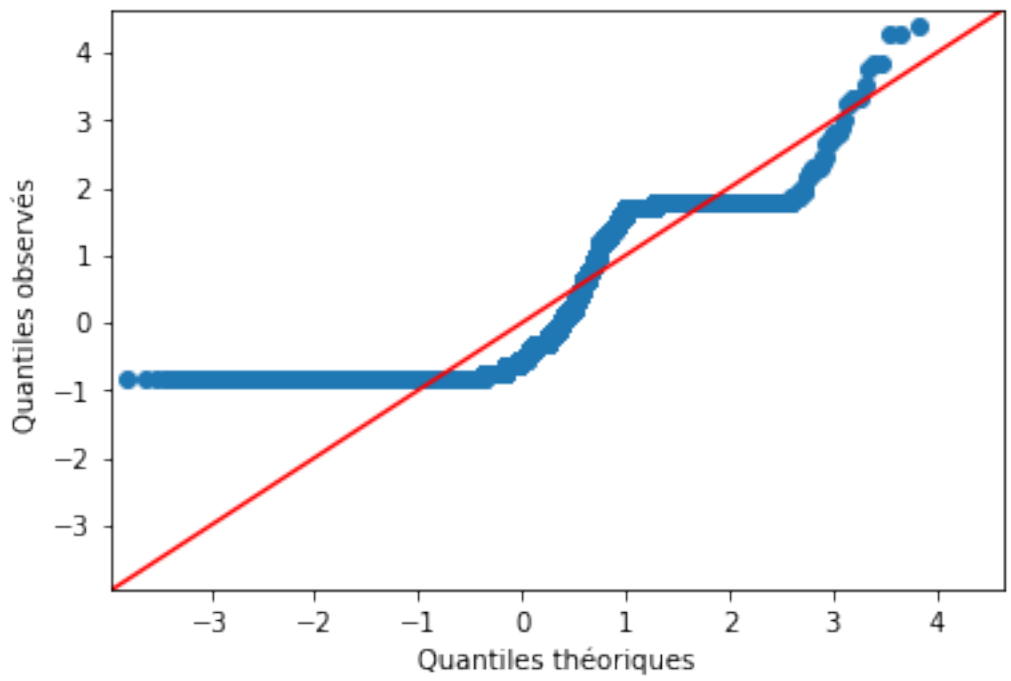
[ ]:
```
mdl_video_vs_genre.params
```

[ ]:
```
Intercept          14.855376
Genre[T.Homme]     -0.988100
dtype: float64
```

[ ]:
```
# Test de normalité de la distribution du nombre de videos (Q-Q plot)
qqplot(data=mdl_video_vs_genre.resid, fit=True, line="45")
plt.show()
```



[ ]:
```
#qqplot(data=etu_genre_video["video"], fit=True, line="45")
qqplot(data=total_video_etu["video"], fit=True, line="45")
plt.xlabel("Quantiles théoriques")
plt.ylabel("Quantiles observés ")
plt.savefig("../graph/distribution_video3.png")
plt.show()
```



[ ]:
```
# Test de  Kolmogrov-Smirnov
stat, p = kstest(total_video_etu["video"], 'norm')
```

[ ]:
```
stat, p
```

[ ]:
```
(0.5391101414598032, 0.0)
```

[ ]:
```
# Test non paramétrique de Mann-Whitney U
U1, p = mannwhitneyu(df_hom["video"], df_fem["video"])
print("U1", U1)
print("p-value", p)

# Calcul de U2 (d'après la doc de scipy.org)
nx, ny = df_hom["video"].count(), df_fem["video"].count()
U2 = nx*ny - U1
print("U2 = ", U2)
```

```
U1 8200580.5
p-value 0.0003907509304995919
U2 =  8994696.5
```

[ ]:
```
# Nombre de quiz par étudiant
total_quiz_etu = total_quiz.groupby("Student_ID").sum()
```

```
[ ]: # Nombre de Quiz et vidéos par étudiants
video_quiz_etu = total_video_etu.merge(total_quiz_etu, on="Student_ID")
```

```
[ ]: # Test de Pearson
correlation, p_value = pearsonr(video_quiz_etu["quiz"], video_quiz_etu["video"])
```

```
[ ]: correlation, p_value
```

```
[ ]: (0.8036026075037674, 0.0)
```

```
[ ]: # Test de Spearman
correlation, p_value = spearmanr(video_quiz_etu["quiz"],␣
 ↪video_quiz_etu["video"])
```

```
[ ]: correlation, p_value
```

```
[ ]: (0.804511796629543, 0.0)
```

Il y a une forte corrélation (0.8) entre le nombre de videos vues et le nombre de quiz réalisés par un étudiant. La corrélation observée est statistiquement significative (p-value=0).

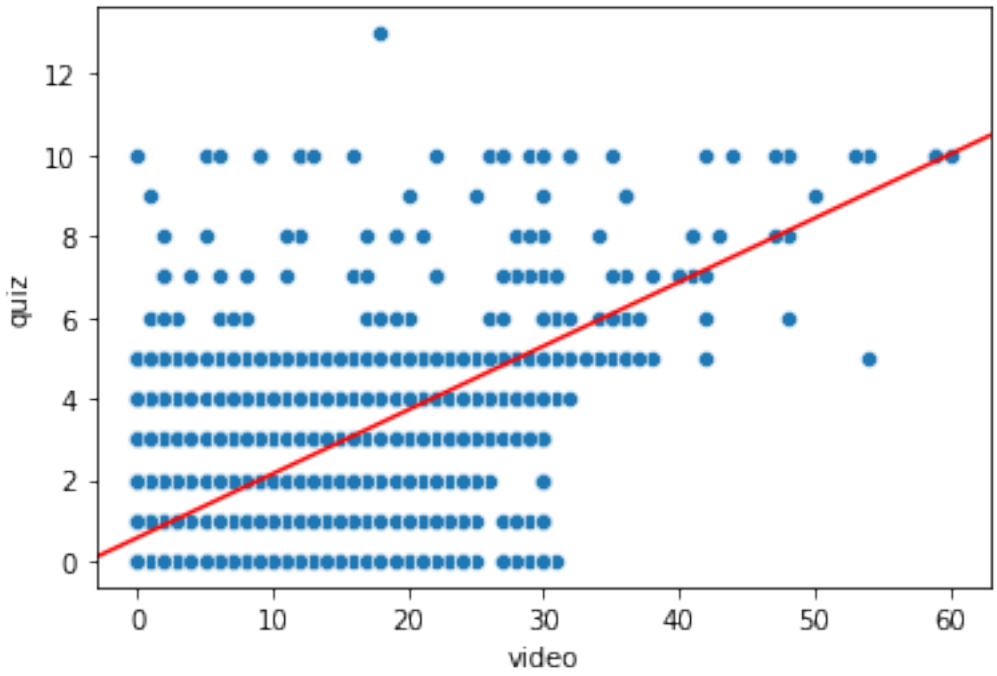
```
[ ]: # Modèle de regression sur le nombre de vidéos selon le nombre de quiz
mdl_video_vs_quiz = ols("quiz ~ video", data=video_quiz_etu).fit()
```

```
[ ]: mdl_video_vs_quiz.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   quiz   R-squared:                       0.646
Model:                            OLS   Adj. R-squared:                  0.646
Method:                 Least Squares   F-statistic:                 2.653e+04
Date:                Sat, 01 Jul 2023   Prob (F-statistic):               0.00
Time:                        08:35:49   Log-Likelihood:                -24981.
No. Observations:               14557   AIC:                            4.997e+04
Df Residuals:                   14555   BIC:                            4.998e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.5652      0.014     39.183      0.000       0.537       0.594
video          0.1575      0.001    162.896      0.000       0.156       0.159
==============================================================================
Omnibus:                     5315.454   Durbin-Watson:                   1.939
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            20116.783
Skew:                           1.821   Prob(JB):                         0.00
```

```
Kurtosis:                       7.461   Cond. No.                         19.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

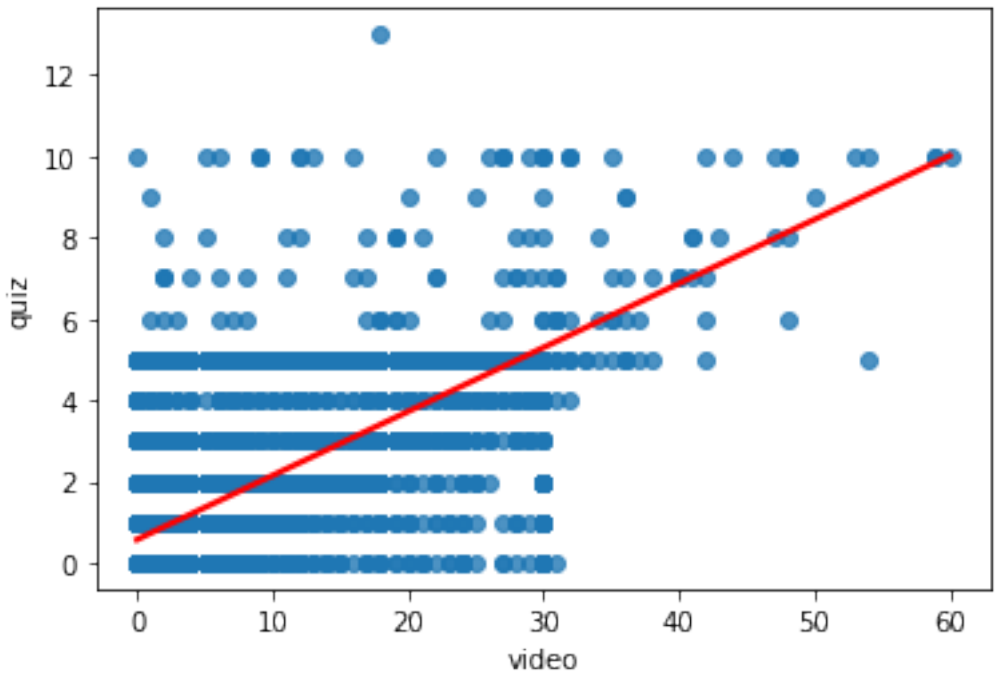
[ ]:
```
intercept, coef = mdl_video_vs_quiz.params
sns.scatterplot(data=video_quiz_etu, x="video", y="quiz")
plt.axline(xy1=(0,intercept), slope=coef, color="red")
plt.savefig("../graph/scatter2_regression.png")
plt.show()
```



[ ]:
```
coef
```

[ ]:
```
0.15750379447434937
```

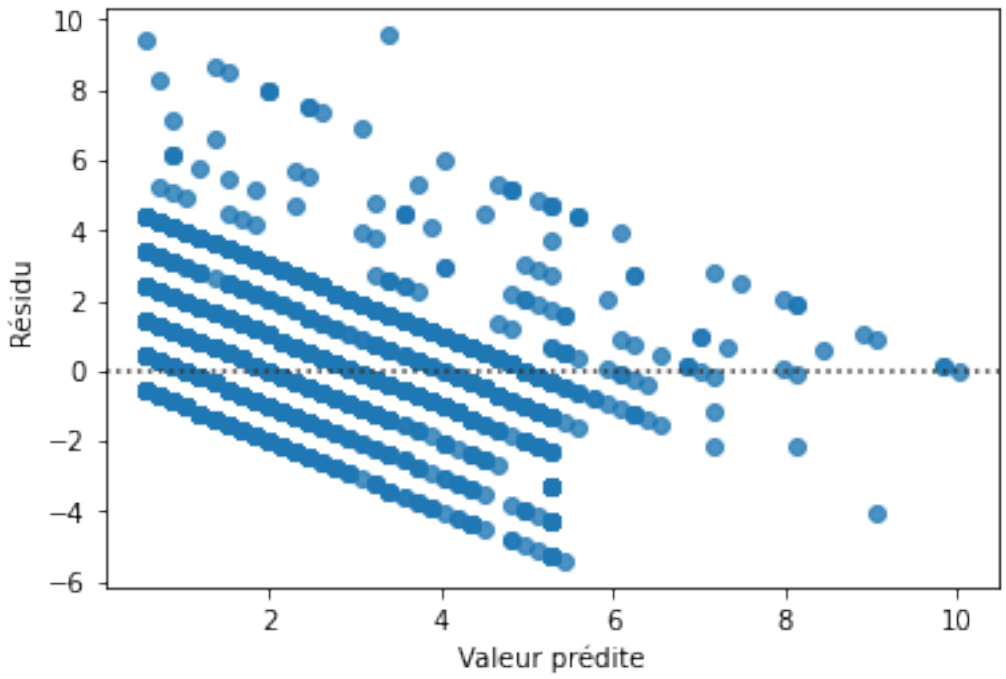
[ ]:
```
fig, ax = plt.subplots()
sns.regplot(data=video_quiz_etu, x=video_quiz_etu["video"],␣
 ↪y=video_quiz_etu["quiz"], line_kws={"color": "red"}, ax=ax)
#plt.savefig("../graph/scatter_regression.png")
plt.show()
```



[ ]:

[ ]:
```
# Calcul du nombre prédit de quiz selon le nombre de videos vues
X = video_quiz_etu[["video"]]
video_quiz_etu["q_predict"] = mdl_video_vs_quiz.predict(X)
```

[ ]:
```
# Résidus du modèle de prédiction
fig, ax = plt.subplots()
sns.residplot(data=video_quiz_etu, x="q_predict", y="quiz", ax=ax)
#ax.set_title("Résidus du modèle observé")
ax.set_ylabel("Résidu")
ax.set_xlabel("Valeur prédite")
plt.savefig("../graph/residus_regression.png")
plt.show()
```



[ ]:
```
# HDI des apprenants
student_hdi = base[["Student_ID", "New_HDI"]].dropna().set_index("Student_ID")
student_hdi.sort_values("Student_ID", inplace=True)
```

[ ]:
```
student_hdi
```

[ ]:
```
            New_HDI
Student_ID
45               TH
83                I
84                B
87               TH
94               TH
…                …
68282             B
68326            TH
68332             I
68365            TH
69565            TH

[8963 rows x 1 columns]
```

[ ]:
```
# Tableau des 3 variables dont 2 catégorielles et 1 continue
video_genre_hdi = etu_genre_video.merge(student_hdi, on="Student_ID")
```

[ ]:
```
video_genre_hdi.head()
```

[ ]:
```
   Student_ID  video  Genre New_HDI
0          45     25  Femme      TH
1          83     22  Homme       I
2          84      8  Homme       B
3          87      1  Homme      TH
4          94      2  Homme      TH
```

[ ]:
```
# Modèle linéaire sans interaction(genre, HDI, video)
mdl1 = ols("video ~ C(Genre) + C(New_HDI)", data=video_genre_hdi).fit()
```

[ ]:
```
mdl1.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  video   R-squared:                       0.057
Model:                            OLS   Adj. R-squared:                  0.056
Method:                 Least Squares   F-statistic:                     177.2
Date:                Sat, 01 Jul 2023   Prob (F-statistic):          1.41e-111
Time:                        08:35:50   Log-Likelihood:                -34351.
No. Observations:                8865   AIC:                         6.871e+04
Df Residuals:                    8861   BIC:                         6.874e+04
Df Model:                           3
Covariance Type:            nonrobust
===============================================================================
=====
                      coef    std err          t      P>|t|      [0.025
0.975]
-------------------------------------------------------------------------------
-----
Intercept             7.0011      0.432     16.211      0.000       6.155
7.848
C(Genre)[T.Homme]    -0.0959      0.267     -0.360      0.719      -0.618
0.427
C(New_HDI)[T.I]       4.6774      0.586      7.979      0.000       3.528
5.826
C(New_HDI)[T.TH]      8.6728      0.395     21.937      0.000       7.898
9.448
==============================================================================
Omnibus:                    11867.785   Durbin-Watson:                   1.891
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              639.581
Skew:                           0.218   Prob(JB):                    1.31e-139
Kurtosis:                       1.759   Cond. No.                         8.79
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[ ]:
```
video_genre_hdi.head()
```

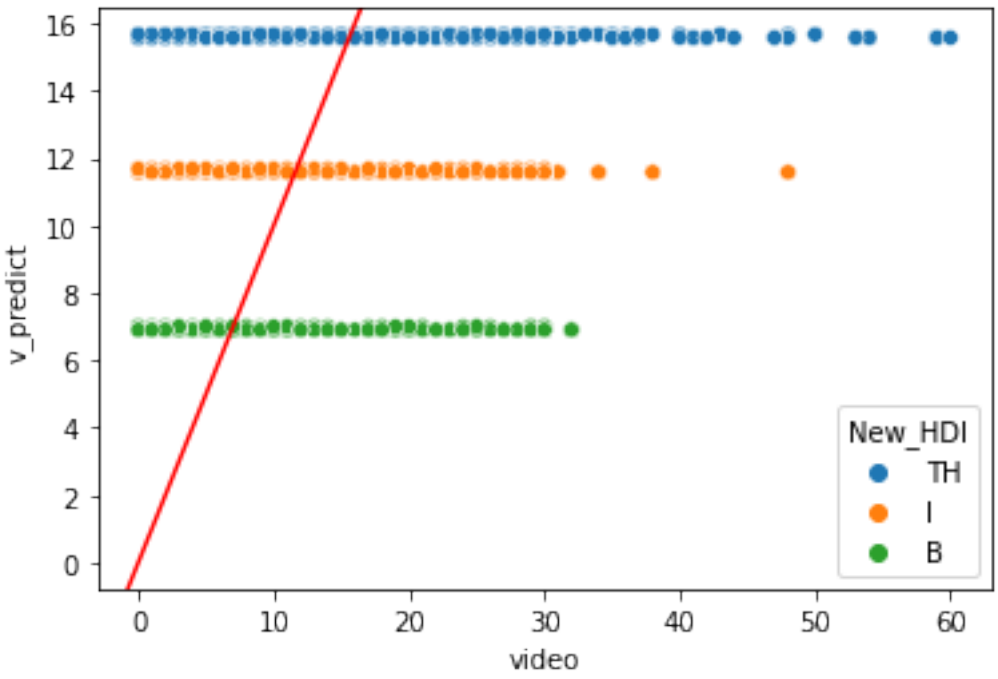
[ ]:
```
   Student_ID  video  Genre New_HDI
0          45     25  Femme      TH
1          83     22  Homme       I
2          84      8  Homme       B
3          87      1  Homme      TH
4          94      2  Homme      TH
```

[ ]:
```
# Calcul du nombre prédit de vidéos
X = video_genre_hdi[['Genre', 'New_HDI']]
video_genre_hdi["v_predict"] = mdl1.predict(X)
```

[ ]:
```
video_genre_hdi
```

[ ]:
```
      Student_ID  video  Genre New_HDI  v_predict
0             45     25  Femme      TH  15.673847
1             83     22  Homme       I  11.582629
2             84      8  Homme       B   6.905238
3             87      1  Homme      TH  15.577989
4             94      2  Homme      TH  15.577989
…            …    …     …        …          …
8860       68282      1  Homme       B   6.905238
8861       68326     30  Femme      TH  15.673847
8862       68332      4  Femme       I  11.678487
8863       68365      0  Femme      TH  15.673847
8864       69565      9  Femme      TH  15.673847

[8865 rows x 5 columns]
```

[ ]:
```
# Graphique du modèle de régression linéaire
sns.scatterplot(data=video_genre_hdi, x="video", y="v_predict", hue="New_HDI")
plt.axline(xy1=(0,0), slope=1, color="red")
plt.savefig("../graph/scatter3_regression.png")
plt.show()
```



[ ]:
```
# ANOVA sans interaction
anova_table = sm.stats.anova_lm(mdl1, typ=1)
```

[ ]:
```
anova_table
```

[ ]:

	df	sum_sq	mean_sq	F	PR(>F)
C(Genre)	1.0	1.554895e+03	1554.895102	11.437356	7.229351e-04
C(New_HDI)	2.0	7.071483e+04	35357.416782	260.078874	1.738249e-110
Residual	8861.0	1.204643e+06	135.948823	NaN	NaN

[ ]:
```
# Modèle de regression avec interaction
mdl2 = ols("video ~ C(Genre)*C(New_HDI)", data=video_genre_hdi).fit()
```

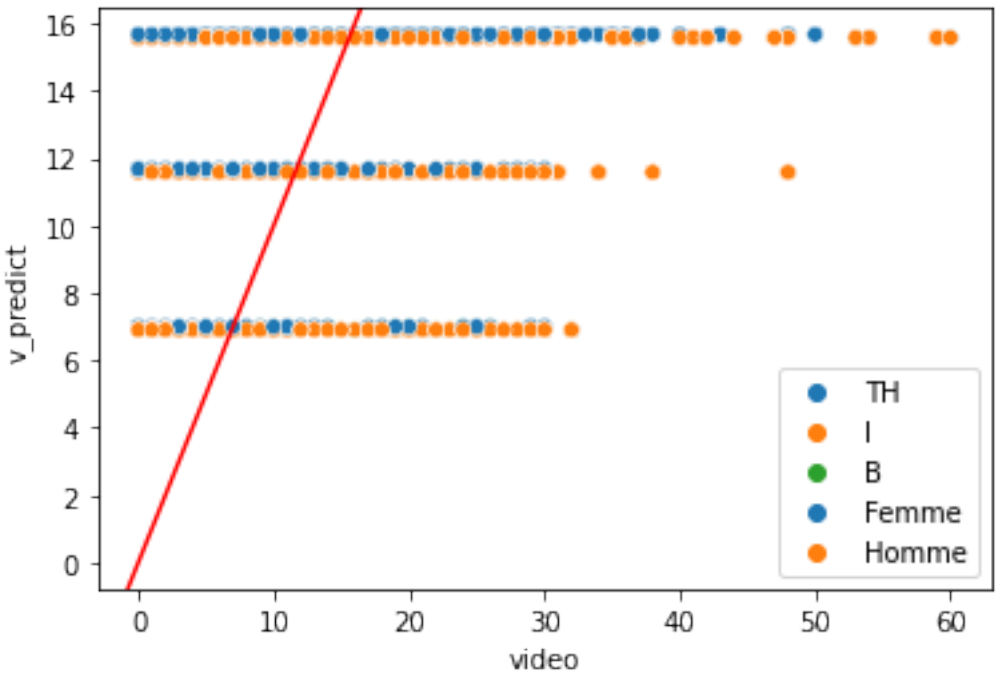
[ ]:
```
mdl2.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  video   R-squared:                       0.057
Model:                            OLS   Adj. R-squared:                  0.057
Method:                 Least Squares   F-statistic:                     107.8
Date:                Sat, 01 Jul 2023   Prob (F-statistic):          7.64e-111
Time:                        08:35:52   Log-Likelihood:                -34347.
No. Observations:                8865   AIC:                         6.871e+04
```

```
Df Residuals:                     8859   BIC:                          6.875e+04
Df Model:                            5
Covariance Type:            nonrobust
==============================================================================================
                                             coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------------------------------
Intercept                                  7.3310      0.968      7.573      0.000
5.434       9.229
C(Genre)[T.Homme]                         -0.4810      1.046     -0.460      0.646
-2.531       1.569
C(New_HDI)[T.I]                            2.7733      1.236      2.244      0.025
0.350       5.196
C(New_HDI)[T.TH]                           8.4673      0.995      8.506      0.000
6.516      10.419
C(Genre)[T.Homme]:C(New_HDI)[T.I]          2.8032      1.415      1.982      0.048
0.030       5.576
C(Genre)[T.Homme]:C(New_HDI)[T.TH]         0.1933      1.085      0.178      0.859
-1.933       2.320
==============================================================================
Omnibus:                    11724.976   Durbin-Watson:                   1.891
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              637.561
Skew:                           0.217   Prob(JB):                    3.59e-139
Kurtosis:                       1.760   Cond. No.                         33.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[ ]:
```
mdl_params = mdl2.params
# Calcul du nombre prédit de vidéos
X = video_genre_hdi[['Genre', 'New_HDI']]
video_genre_hdi["v_predict"] = mdl1.predict(X)
```

[ ]:
```
# Graphique du modèle de régression linéaire
sns.scatterplot(data=video_genre_hdi, x="video", y="v_predict", hue="New_HDI")
sns.scatterplot(data=video_genre_hdi, x="video", y="v_predict", hue="Genre")
plt.axline(xy1=(0,0), slope=1, color="red")
plt.legend()
plt.savefig("../graph/scatter3_regression.png")
plt.show()
```



[ ]:
```
# ANOVA avec interaction
anova_table = sm.stats.anova_lm(mdl2, typ=1)
```

[ ]:
```
anova_table
```

[ ]:
```
                          df         sum_sq       mean_sq           F  \
C(Genre)                 1.0   1.554895e+03   1554.895102   11.443839
C(New_HDI)               2.0   7.071483e+04  35357.416782  260.226287
C(Genre):C(New_HDI)      2.0   9.541531e+02    477.076535    3.511225
Residual              8859.0   1.203688e+06    135.871810         NaN

                            PR(>F)
C(Genre)                7.204212e-04
C(New_HDI)             1.514733e-110
C(Genre):C(New_HDI)     2.990187e-02
Residual                         NaN
```

[ ]:
```
df_exam.rename(columns={"Exam.bin": "Exam", "Certif.bin": "Certif"},␣
 ↪inplace=True)
df_exam.reset_index(inplace=True)
df_exam.dropna(subset=["Gender", "New_HDI"], inplace=True)
```

[ ]:
```
df_exam
```

[ ]:

	Itération	Student_ID	Gender	New_HDI	Exam	Certif	Assignment.bin
1	1	19178	une femme	TH	False	False	0
2	1	1086	une femme	TH	False	False	0
3	1	1948	une femme	TH	False	False	0
4	1	16209	une femme	B	False	False	0
8	1	402	une femme	B	False	False	1
…	…	…	…	…	…	…	…
15165	3	33473	une femme	TH	False	False	0
15167	3	64940	un homme	TH	False	False	0
15172	3	33848	un homme	TH	False	False	0
15176	3	24513	un homme	TH	False	False	0
15177	3	42092	un homme	B	False	False	1

[8951 rows x 7 columns]

[ ]:
```
# Obtention de l'examen et/ou de la certification
df_exam["Exam_Certif"] = df_exam["Exam"] | df_exam["Certif"]
df_exam["Exam_Certif"] = df_exam["Exam_Certif"].astype(int)
```

[ ]:
```
df_exam.set_index("Itération", inplace=True)
```

[ ]:
```
# Obtention ou non par itération
df_exam_v1 = df_exam.loc[1]
df_exam_v2 = df_exam.loc[2]
df_exam_v3 = df_exam.loc[3]
```

[ ]:
```
df_exam_v2
```

[ ]:

Itération	Student_ID	Gender	New_HDI	Exam	Certif	Assignment.bin	Exam_Certif
2	32360	une femme	TH	True	False	1	1
2	27808	un homme	B	True	False	1	1
2	27532	un homme	TH	True	False	1	1
2	2630	un homme	B	True	False	1	1
2	23971	un homme	TH	False	False	0	0
…	…	…	…	…	…	…	…
2	29275	une femme	TH	True	False	1	1
2	28828	un homme	I	True	False	1	1
2	26940	un homme	TH	True	False	1	1
2	28699	une femme	I	False	False	0	0
2	27897	un homme	TH	False	False	0	0

```
2                    1
2                    0
…                    …
2                    1
2                    1
2                    1
2                    0
2                    0

[2154 rows x 7 columns]
```

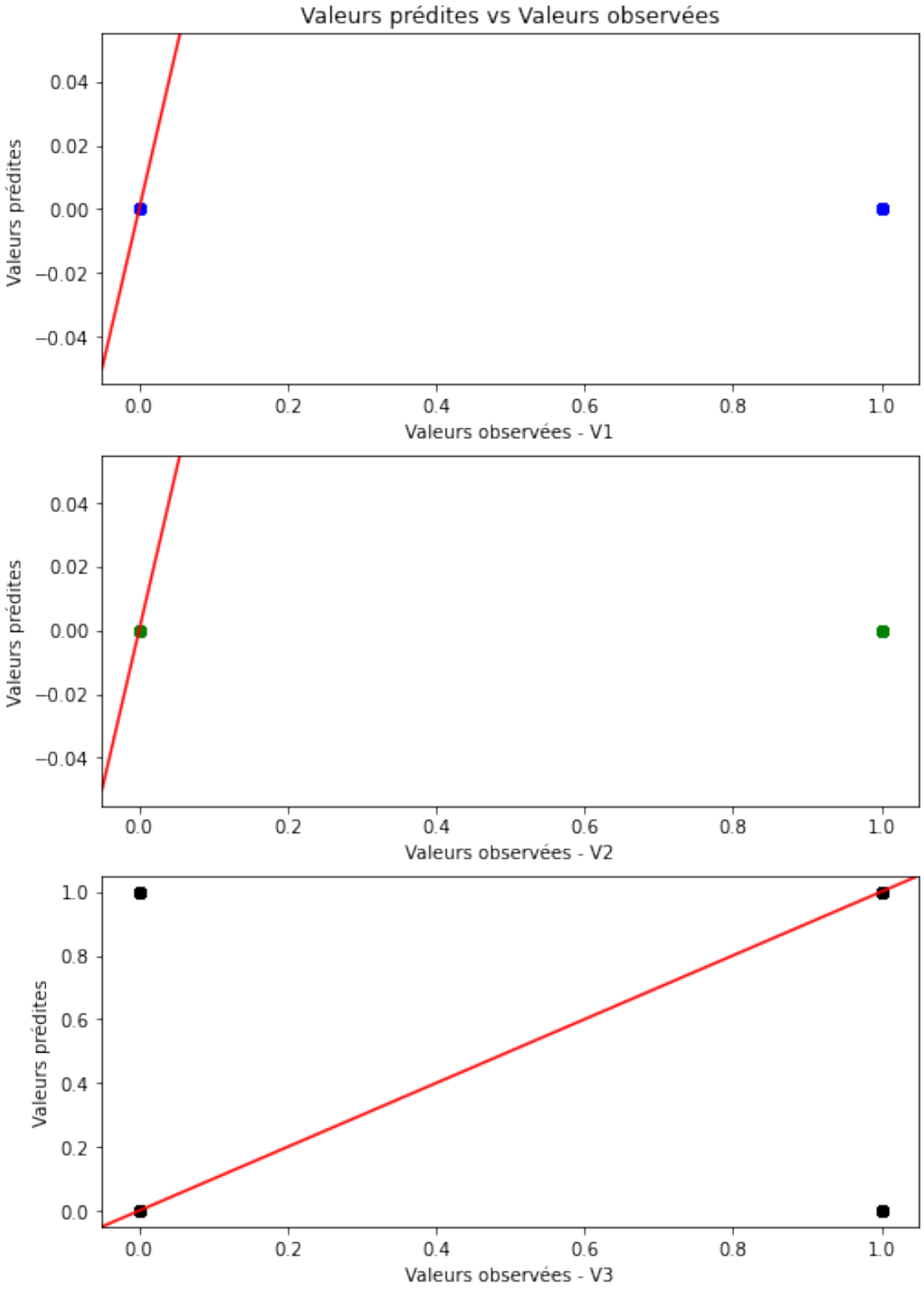
[ ]:
```
# Modèle logistic (GLM) de type binomial (variable dépendante binaire) appliqué␣
 ↪pour chaque itération
formula = "Exam_Certif ~ Gender + New_HDI"
model = []
for n, version in enumerate([df_exam_v1, df_exam_v2, df_exam_v3]):
    model.append(glm(formula=formula, data=version, family=sm.families.
 ↪Binomial()))
```

[ ]:
```
# résultats de la fonction logistic par itération
result_v1 = model[0].fit()
result_v2 = model[1].fit()
result_v3 = model[2].fit()
```

[ ]:
```
# Valeurs prédites pour chaque itérations
df_exam_v1["predicted"] = np.round(result_v1.predict(), decimals=0).astype(int)
df_exam_v2["predicted"] = np.round(result_v2.predict(), decimals=0).astype(int)
df_exam_v3["predicted"] = np.round(result_v3.predict(), decimals=0).astype(int)
```

[ ]:
```
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(8,12))
ax1.scatter(x=df_exam_v1["Exam_Certif"], y=df_exam_v1['predicted'],␣
 ↪color="blue")
ax1.axline((0, 0), slope=1, color='red', linestyle='-')
ax1.set_xlabel('Valeurs observées - V1')
ax1.set_ylabel('Valeurs prédites')
ax1.set_title('Valeurs prédites vs Valeurs observées')

ax2.scatter(x=df_exam_v2["Exam_Certif"], y=df_exam_v2['predicted'],␣
 ↪color="green")
ax2.axline((0, 0), slope=1, color='red', linestyle='-')
ax2.set_xlabel('Valeurs observées - V2')
ax2.set_ylabel('Valeurs prédites')

ax3.scatter(x=df_exam_v3["Exam_Certif"], y=df_exam_v3['predicted'],␣
 ↪color="black")
ax3.axline((0, 0), slope=1, color='red', linestyle='-')
ax3.set_xlabel('Valeurs observées - V3')
```

```
ax3.set_ylabel('Valeurs prédites')

plt.show()
```



[ ]:
```
result_v1.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:            Exam_Certif   No. Observations:                 5237
Model:                            GLM   Df Residuals:                     5233
Model Family:                Binomial   Df Model:                           3
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -128.64
Date:                Sat, 01 Jul 2023   Deviance:                       257.28
Time:                        08:35:54   Pearson chi2:                 4.84e+03
No. Iterations:                    25   Pseudo R-squ. (CS):          0.001021
Covariance Type:            nonrobust
==============================================================================
=======
                           coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept               -5.9120      0.715     -8.266      0.000      -7.314
-4.510
Gender[T.une femme]      0.6235      0.456      1.367      0.172      -0.270
1.517
New_HDI[T.I]          -20.8950   1.75e+04     -0.001      0.999   -3.43e+04
3.42e+04
New_HDI[T.TH]           0.2241      0.756      0.297      0.767      -1.257
1.705
==============================================================================
=======
"""
```

[ ]:
```
# Calcul des odds-ratio, p-value et ci pour chaque itération par rapport à␣
 ↪l'intercept
mooc = ["V1", "V2", "V3"]
full_odds_r, full_pvalues, full_ci = [], [], []
for n, version in enumerate([result_v1, result_v2, result_v3]):
    # OR
    odds_r = np.exp(version.params).round(3)
    odds_r["Intercept"] = 1
    odds_r.name = mooc[n]
    full_odds_r.append(odds_r)
    # p-value
```

```
    p_values = version.pvalues.round(3)
    p_values.name = mooc[n]
    full_pvalues.append(p_values)
    # interval de confiance (ci)
    ci = np.exp(version.conf_int()) # intervalle à 95 %
    ci = "[" + ci[0].round(3).astype(str) + ", " + ci[1].round(3).astype(str) +␣
 ↪"]"
    ci.name = mooc[n]
    full_ci.append(ci)
```

[ ]:
```
tab_coef = pd.concat([pd.DataFrame(full_odds_r).T, pd.DataFrame(full_pvalues).
 ↪T, pd.DataFrame(full_ci).T])
```

[ ]:
```
index = pd.MultiIndex.from_arrays([["odds-ratio", "odds-ratio", "odds-ratio",␣
 ↪"odds-ratio",
                                    "p-value", "p-value", "p-value", "p-value",
                                    "ci", "ci", "ci", "ci"],
                                    tab_coef.index], names=["type", "coef"])
```

[ ]:
```
tab_full = pd.DataFrame({"V1": list(tab_coef["V1"]), "V2":␣
 ↪list(tab_coef["V2"]), "V3": list(tab_coef["V3"])}, index=index)
```

[ ]:
```
tab_full.reset_index(inplace=True)
```

[ ]:
```
tab_full.set_index(["type"], inplace=True)
```

[ ]:
```
tab_full
```

[ ]:
```
                          coef              V1              V2  \
type
odds-ratio           Intercept             1.0             1.0
odds-ratio  Gender[T.une femme]           1.865           1.021
odds-ratio         New_HDI[T.I]             0.0           1.046
odds-ratio        New_HDI[T.TH]           1.251           1.558
p-value              Intercept             0.0             0.0
p-value     Gender[T.une femme]           0.172           0.826
p-value            New_HDI[T.I]           0.999           0.849
p-value           New_HDI[T.TH]           0.767           0.004
ci                   Intercept  [0.001, 0.011]  [0.329, 0.587]
ci          Gender[T.une femme]   [0.763, 4.56]  [0.851, 1.224]
ci                 New_HDI[T.I]      [0.0, inf]   [0.66, 1.658]
ci                New_HDI[T.TH]  [0.284, 5.503]  [1.149, 2.113]

                           V3
type
odds-ratio            1.0
odds-ratio          0.813
```

```
odds-ratio           1.062
odds-ratio           0.835
p-value              0.157
p-value              0.053
p-value              0.813
p-value              0.311
ci          [0.913, 1.762]
ci           [0.66, 1.003]
ci          [0.647, 1.741]
ci          [0.589, 1.184]
```

p > 0.05 : Non significatif (pas d'astérisque) 0.01 < p 0.05 : * (un astérisque) 0.001 < p 0.01 : ** (deux astérisques) p 0.001 : *** (trois astérisques)

[ ]:
```
ci_v1 = tab_full.loc["ci", ["coef", "V1"]]
or_v1 = tab_full.loc["odds-ratio", ["coef", "V1"]]
pval_v1 = tab_full.loc["p-value", ["coef", "V1"]]

ci_v2 = tab_full.loc["ci", ["coef", "V2"]]
or_v2 = tab_full.loc["odds-ratio", ["coef", "V2"]]
pval_v2 = tab_full.loc["p-value", ["coef", "V2"]]

ci_v3 = tab_full.loc["ci", ["coef", "V3"]]
or_v3 = tab_full.loc["odds-ratio", ["coef", "V3"]]
pval_v3 = tab_full.loc["p-value", ["coef", "V3"]]
```

[ ]:
```
ci_v1.rename(columns={'V1': 'CI'}, inplace=True)
ci_v2.rename(columns={'V2': 'CI'}, inplace=True)
ci_v3.rename(columns={'V3': 'CI'}, inplace=True)

or_v1.rename(columns={'V1': 'OR'}, inplace=True)
or_v2.rename(columns={'V2': 'OR'}, inplace=True)
or_v3.rename(columns={'V3': 'OR'}, inplace=True)

pval_v1.rename(columns={'V1': 'pval'}, inplace=True)
pval_v2.rename(columns={'V2': 'pval'}, inplace=True)
pval_v3.rename(columns={'V3': 'pval'}, inplace=True)
```

[ ]:
```
# Extraction des bornes inférieure et supérieure des CI
ci_v1[["ll", "hl"]] = ci_v1["CI"].str.strip("[]").str.split(",", expand=True).
 ↪astype(float)
ci_v2[["ll", "hl"]] = ci_v2["CI"].str.strip("[]").str.split(",", expand=True).
 ↪astype(float)
ci_v3[["ll", "hl"]] = ci_v3["CI"].str.strip("[]").str.split(",", expand=True).
 ↪astype(float)
```

[ ]:
```
# Tableaux des valeurs pour créer le forestplot
forest_v1 = ci_v1.merge(or_v1, on="coef").merge(pval_v1, on="coef")
forest_v2 = ci_v2.merge(or_v2, on="coef").merge(pval_v2, on="coef")
forest_v3 = ci_v3.merge(or_v3, on="coef").merge(pval_v3, on="coef")
```

[ ]:
```
# Renommage de la référence "un homme"
forest_v1.iloc[0, 0] = "Réf homme"
forest_v2.iloc[0, 0] = "Réf homme"
forest_v3.iloc[0, 0] = "Réf homme"

# Ajout de la 2eme référence (HDI B)
full_pvalues = pd.DataFrame(full_pvalues)
forest_v1 = forest_v1.append(pd.Series(["Réf HDI B", "[0,0]", 0, 0, 1, ""],␣
 ↪index=forest_v1.columns), ignore_index=True)
forest_v2 = forest_v2.append(pd.Series(["Réf HDI B", "[0,0]", 0, 0, 1, ""],␣
 ↪index=forest_v2.columns), ignore_index=True)
forest_v3 = forest_v3.append(pd.Series(["Réf HDI B", "[0,0]", 0, 0, 1, ""],␣
 ↪index=forest_v3.columns), ignore_index=True)
```

[ ]:
```
forest_v1
```

[ ]:

	coef	CI	ll	hl	OR	pval
0	Réf homme	[0.001, 0.011]	0.001	0.011	1.0	0.0
1	Gender[T.une femme]	[0.763, 4.56]	0.763	4.560	1.865	0.172
2	New_HDI[T.I]	[0.0, inf]	0.000	inf	0.0	0.999
3	New_HDI[T.TH]	[0.284, 5.503]	0.284	5.503	1.251	0.767
4	Réf HDI B	[0,0]	0.000	0.000	1	

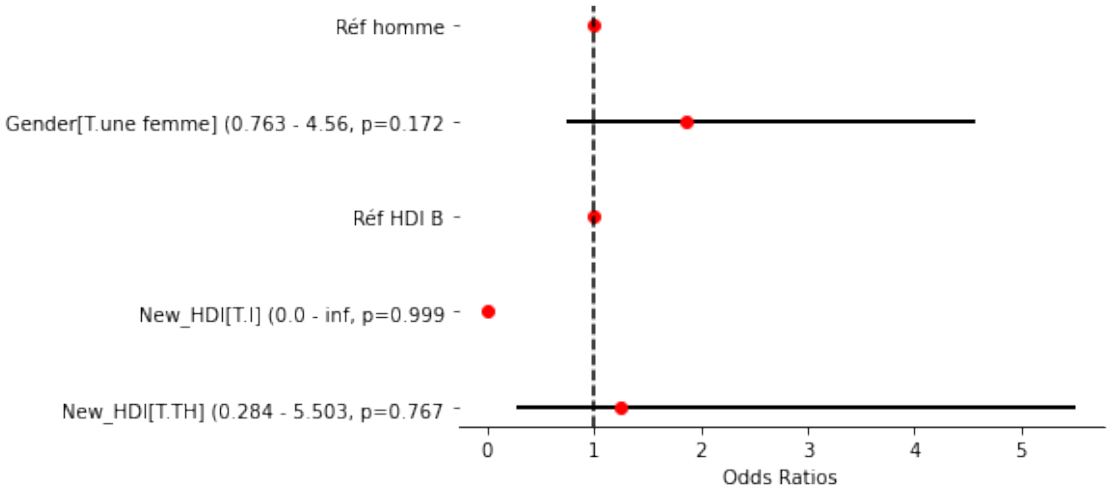
[ ]:
```
forest = []
for v in [forest_v1, forest_v2, forest_v3]:
    v.iloc[[2, 4]] = v.iloc[[4, 2]]
    v.iloc[[3, 4]] = v.iloc[[4, 3]]
    # Mise à zero des intervalles de confiance de l'intercept
    v.iloc[0, 2] = 0
    v.iloc[0, 3] = 0
    v.iloc[2, 2] = 0
    v.iloc[2, 3] = 0
    forest.append(v)
```

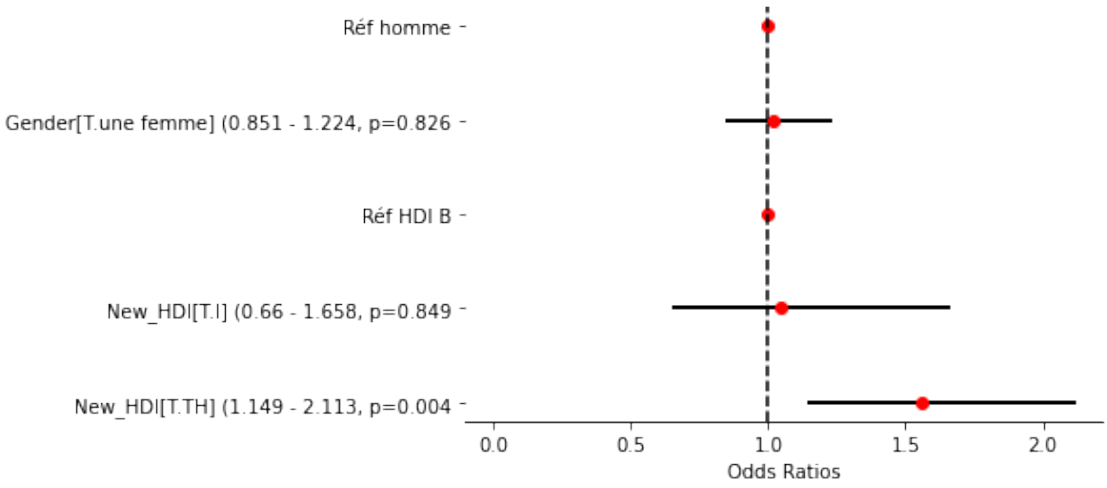
[ ]:
```
forest[2]
```

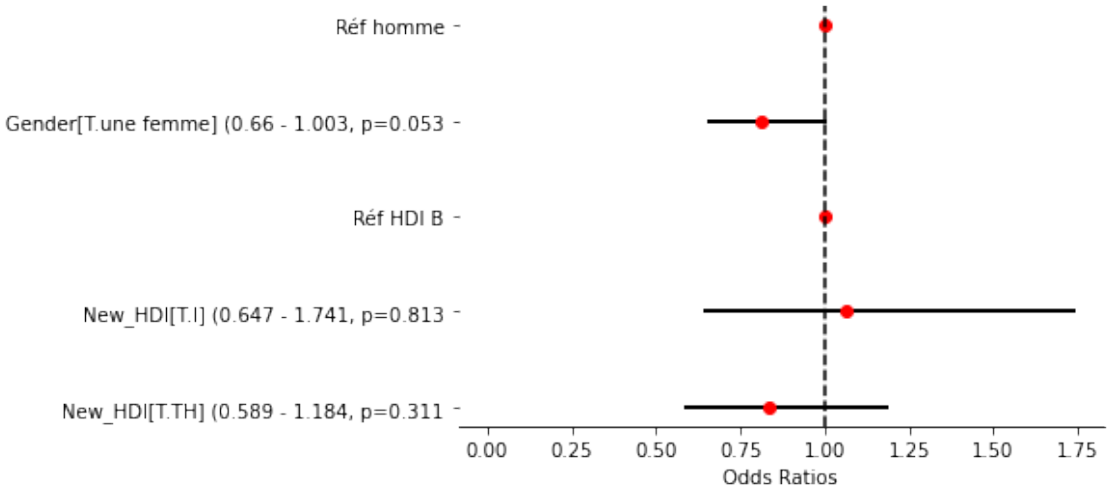
[ ]:

	coef	CI	ll	hl	OR	pval
0	Réf homme	[0.913, 1.762]	0.000	0.000	1.0	0.157
1	Gender[T.une femme]	[0.66, 1.003]	0.660	1.003	0.813	0.053
2	Réf HDI B	[0,0]	0.000	0.000	1	
3	New_HDI[T.I]	[0.647, 1.741]	0.647	1.741	1.062	0.813
4	New_HDI[T.TH]	[0.589, 1.184]	0.589	1.184	0.835	0.311

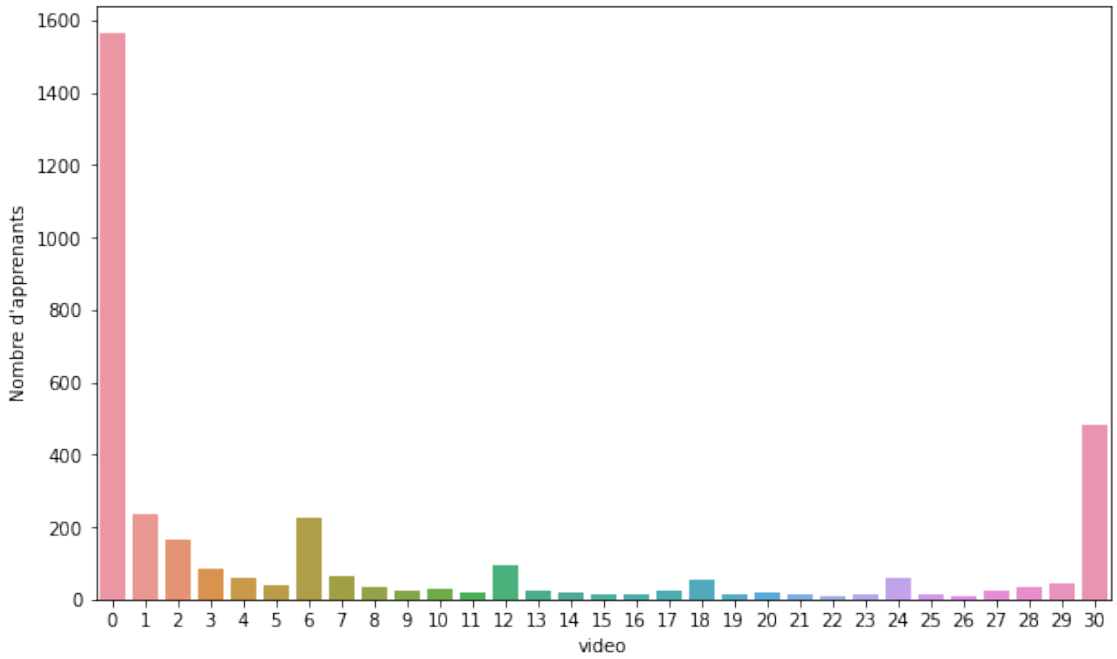
[ ]:
```
for n, version in enumerate(forest):
    version = version[::-1]
    # Données de la régression logistique
    odds_ratios = version["OR"].tolist()
    ci = list(zip(version["ll"], version["hl"]))
    ci_inf = version["ll"].astype(str)
    ci_sup = version["hl"].astype(str)
    labels = version["coef"] + " (" + ci_inf + " - " + ci_sup + ", p=" +␣
 ↪version["pval"].astype(str)
    labels.loc[0] = "Réf homme"
    labels.loc[2] = "Réf HDI B"

    # Création du forest plot
    fig, ax = plt.subplots()
    # OR, intervalles de confiance
    for i, (or_val, (ci_low, ci_high), label) in enumerate(zip(odds_ratios, ci,␣
 ↪labels)):
        ax.plot([ci_low, ci_high], [i, i], color="black", linewidth=2)
        ax.plot(or_val, i, 'ro')

    # Réglages des axes
    ax.set_xlim()
    ax.set_yticks(range(len(labels)))
    ax.set_yticklabels(labels)
    ax.set_xlabel('Odds Ratios')
    ax.axvline(1, color='black', linestyle='--')
    ax.spines["top"].set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["right"].set_visible(False)
    file_name = "../graph/forestplot_V" + str(n) + "_binomiale.png"
    #plt.savefig(file_name)
    plt.show()
```







[ ]:
```
# Distribution du nombre de videos pour l'ensemble des versions
fig, ax = plt.subplots(figsize=(10,6))
sns.countplot(data=total_video.loc[3], x="video", ax=ax)
plt.ylabel("Nombre d'apprenants")
plt.savefig("../graph/distribution_poisson.png")
plt.show()
```



[ ]:
```
total_video
```

[ ]:
```
                                        video
Itération Student_ID Gender    New_HDI
1         221        NaN       NaN          0
          19178      une femme TH           1
          1086       une femme TH          30
          1948       une femme TH           1
          16209      une femme B            0
…                                       …
3         42092      un homme  B            2
          64673      NaN       NaN          0
          67894      NaN       NaN          0
          66874      NaN       NaN          1
          66492      NaN       NaN          0

[15182 rows x 1 columns]
```

[ ]:
```
df_glm_video = total_video.reset_index(["Gender", "New_HDI"]).dropna()
```

[ ]:
```
df_glm_video
```

[ ]:
```
                          Gender New_HDI  video
Itération Student_ID
1         19178       une femme      TH      1
```

```
            1086        une femme      TH     30
            1948        une femme      TH      1
            16209       une femme       B      0
            402         une femme       B     11
…                           …     …     …
3           33473       une femme      TH      6
            64940        un homme      TH      0
            33848        un homme      TH     18
            24513        un homme      TH     18
            42092        un homme       B      2

[8951 rows x 3 columns]
```

[ ]:
```
df_video_v1 = df_glm_video.loc[1]
df_video_v2 = df_glm_video.loc[2]
df_video_v3 = df_glm_video.loc[3]
```

[ ]:
```
df_video_v3
```

[ ]:
```
                Gender New_HDI  video
Student_ID
68029         un homme      TH     30
66198         un homme       I      1
68052         un homme       I      1
14161         un homme       B      4
64444         un homme       B      0
…                  …     …     …
33473       une femme      TH      6
64940         un homme      TH      0
33848         un homme      TH     18
24513         un homme      TH     18
42092         un homme       B      2

[1560 rows x 3 columns]
```

[ ]:
```
formula = "video ~ C(Gender) + C(New_HDI)"
# Regression de Poisson
model = []
for n, version in enumerate([df_video_v1, df_video_v2, df_video_v3]):
    model.append(glm(formula=formula, data=version, family=sm.families.
 ↪Poisson()))
```

[ ]:
```
# résultats de la fonction logistic par itération
result_v1 = model[0].fit()
result_v2 = model[1].fit()
result_v3 = model[2].fit()
```

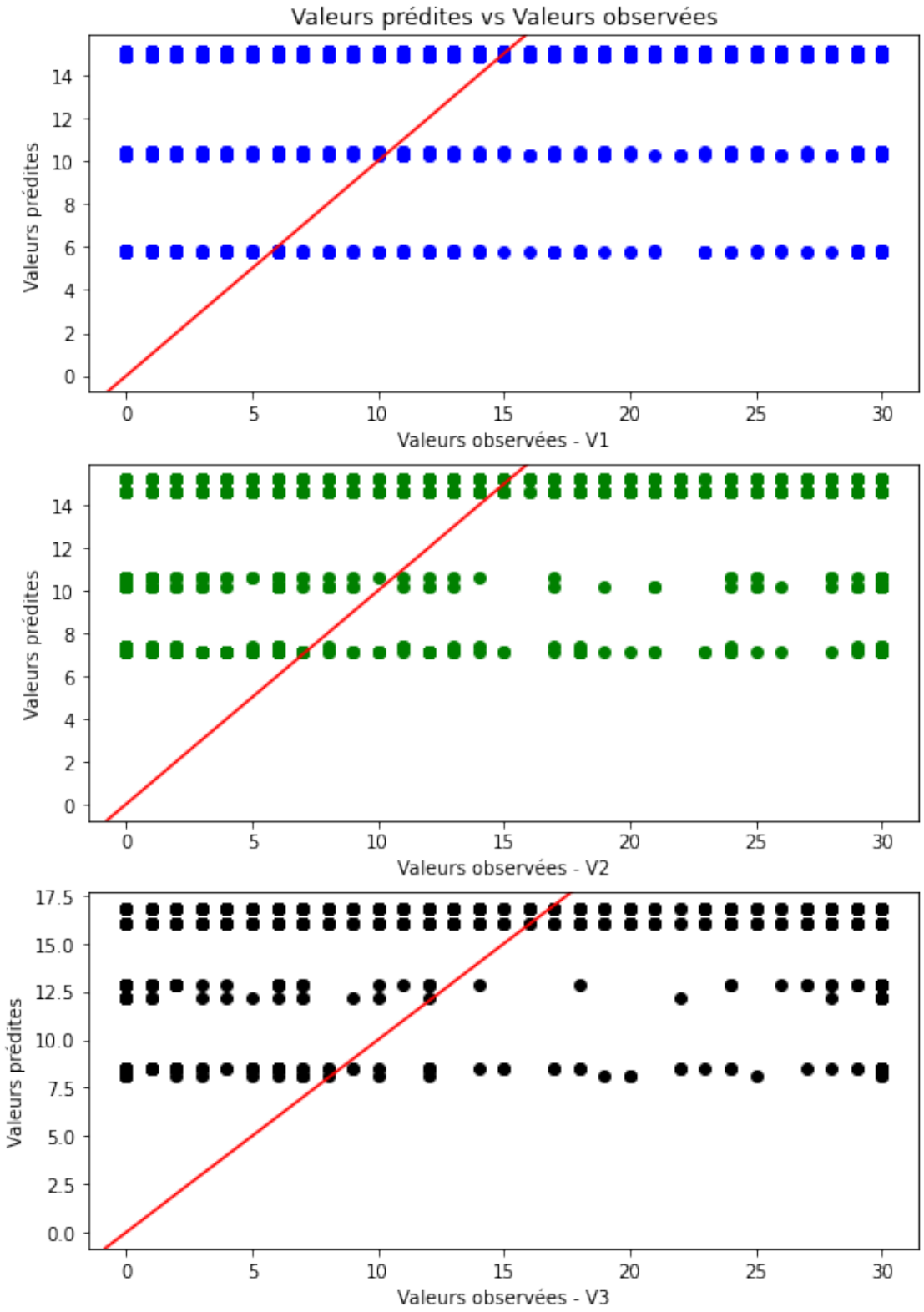
[ ]:
```
result_v3.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  video   No. Observations:                 1560
Model:                            GLM   Df Residuals:                     1556
Model Family:                 Poisson   Df Model:                            3
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -11729.
Date:                Sat, 01 Jul 2023   Deviance:                       17520.
Time:                        08:35:56   Pearson chi2:                 1.44e+04
No. Iterations:                     5   Pseudo R-squ. (CS):             0.3706
Covariance Type:            nonrobust
==============================================================================
==========
                               coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
Intercept                    2.1441      0.029     75.004      0.000       2.088
2.200
C(Gender)[T.une femme]      -0.0484      0.013     -3.588      0.000      -0.075
-0.022
C(New_HDI)[T.I]              0.4085      0.039     10.518      0.000       0.332
0.485
C(New_HDI)[T.TH]             0.6808      0.029     23.080      0.000       0.623
0.739
==============================================================================
==========
"""
```

[ ]:
```
# Valeurs prédites pour chaque itérations
df_video_v1["predicted"] = result_v1.predict()
df_video_v2["predicted"] = result_v2.predict()
df_video_v3["predicted"] = result_v3.predict()
```

[ ]:
```
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(8,12))
ax1.scatter(x=df_video_v1["video"], y=df_video_v1['predicted'], color="blue")
ax1.axline((0, 0), slope=1, color='red', linestyle='-')
ax1.set_xlabel('Valeurs observées - V1')
ax1.set_ylabel('Valeurs prédites')
ax1.set_title('Valeurs prédites vs Valeurs observées')

ax2.scatter(x=df_video_v2["video"], y=df_video_v2['predicted'], color="green")
ax2.axline((0, 0), slope=1, color='red', linestyle='-')
```

```
ax2.set_xlabel('Valeurs observées - V2')
ax2.set_ylabel('Valeurs prédites')

ax3.scatter(x=df_video_v3["video"], y=df_video_v3['predicted'], color="black")
ax3.axline((0, 0), slope=1, color='red', linestyle='-')
ax3.set_xlabel('Valeurs observées - V3')
ax3.set_ylabel('Valeurs prédites')

plt.show()
```


Valeurs prédites vs Valeurs observées
Valeurs prédites
Valeurs observées - V1
Valeurs prédites
Valeurs observées - V2
Valeurs prédites
Valeurs observées - V3

[ ]:
```
# Calcul des odds-ratio, p-value et ci pour chaque itération
full_odds_r, full_pvalues, full_ci = [], [], []
for n, version in enumerate([result_v1, result_v2, result_v3]):
    # OR
    odds_r = np.exp(version.params).round(3)
    odds_r["Intercept"] = 1
    odds_r.name = mooc[n]
    full_odds_r.append(odds_r)
    # p-value
    p_values = version.pvalues.round(3)
    p_values.name = mooc[n]
    full_pvalues.append(p_values)
    # intervalles de confiance (ci)
    ci = np.exp(version.conf_int(alpha=0.05)) # intervalle à 95 %
    ci = "[" + ci[0].round(3).astype(str) + ", " + ci[1].round(3).astype(str) +␣
 ↪"]"
    ci.name = mooc[n]
    full_ci.append(ci)
```

[ ]:
```
tab_coef = pd.concat([pd.DataFrame(full_odds_r).T, pd.DataFrame(full_pvalues).
 ↪T, pd.DataFrame(full_ci).T])
```

[ ]:
```
tab_coef
```

[ ]:

	V1	V2	V3
Intercept	1.0	1.0	1.0
C(Gender)[T.une femme]	1.018	1.039	0.953
C(New_HDI)[T.I]	1.784	1.427	1.505
C(New_HDI)[T.TH]	2.577	2.047	1.975
Intercept	0.0	0.0	0.0
C(Gender)[T.une femme]	0.032	0.002	0.0
C(New_HDI)[T.I]	0.0	0.0	0.0
C(New_HDI)[T.TH]	0.0	0.0	0.0
Intercept	[5.58, 5.945]	[6.806, 7.515]	[8.069, 9.026]
C(Gender)[T.une femme]	[1.001, 1.034]	[1.015, 1.064]	[0.928, 0.978]
C(New_HDI)[T.I]	[1.708, 1.864]	[1.328, 1.532]	[1.394, 1.624]
C(New_HDI)[T.TH]	[2.494, 2.663]	[1.945, 2.154]	[1.864, 2.093]

[ ]:
```
index = pd.MultiIndex.from_arrays([["odds-ratio", "odds-ratio", "odds-ratio",␣
 ↪"odds-ratio",
                                    "p-value", "p-value", "p-value", "p-value",
                                    "ci", "ci", "ci", "ci"],
                                    tab_coef.index], names=["type", "coef"])
```

[ ]:
```
tab_full = pd.DataFrame({"V1": list(tab_coef["V1"]), "V2":␣
 ↪list(tab_coef["V2"]), "V3": list(tab_coef["V3"])}, index=index)
tab_full.reset_index(inplace=True)
```

```
tab_full.set_index(["type"], inplace=True)
```

[ ]:
```
tab_full
```

[ ]:
```
                                coef              V1              V2  \
type
odds-ratio                 Intercept             1.0             1.0
odds-ratio  C(Gender)[T.une femme]           1.018           1.039
odds-ratio         C(New_HDI)[T.I]           1.784           1.427
odds-ratio        C(New_HDI)[T.TH]           2.577           2.047
p-value                    Intercept             0.0             0.0
p-value     C(Gender)[T.une femme]           0.032           0.002
p-value            C(New_HDI)[T.I]             0.0             0.0
p-value           C(New_HDI)[T.TH]             0.0             0.0
ci                         Intercept   [5.58, 5.945]  [6.806, 7.515]
ci          C(Gender)[T.une femme]   [1.001, 1.034]  [1.015, 1.064]
ci                 C(New_HDI)[T.I]   [1.708, 1.864]  [1.328, 1.532]
ci                C(New_HDI)[T.TH]   [2.494, 2.663]  [1.945, 2.154]

                             V3
type
odds-ratio              1.0
odds-ratio            0.953
odds-ratio            1.505
odds-ratio            1.975
p-value                 0.0
p-value                 0.0
p-value                 0.0
p-value                 0.0
ci            [8.069, 9.026]
ci            [0.928, 0.978]
ci            [1.394, 1.624]
ci            [1.864, 2.093]
```

[ ]:
```
ci_v1 = tab_full.loc["ci", ["coef", "V1"]]
or_v1 = tab_full.loc["odds-ratio", ["coef", "V1"]]
pval_v1 = tab_full.loc["p-value", ["coef", "V1"]]

ci_v2 = tab_full.loc["ci", ["coef", "V2"]]
or_v2 = tab_full.loc["odds-ratio", ["coef", "V2"]]
pval_v2 = tab_full.loc["p-value", ["coef", "V2"]]

ci_v3 = tab_full.loc["ci", ["coef", "V3"]]
or_v3 = tab_full.loc["odds-ratio", ["coef", "V3"]]
pval_v3 = tab_full.loc["p-value", ["coef", "V3"]]
```

[ ]:
```
ci_v1.rename(columns={'V1': 'CI'}, inplace=True)
ci_v2.rename(columns={'V2': 'CI'}, inplace=True)
ci_v3.rename(columns={'V3': 'CI'}, inplace=True)

or_v1.rename(columns={'V1': 'OR'}, inplace=True)
or_v2.rename(columns={'V2': 'OR'}, inplace=True)
or_v3.rename(columns={'V3': 'OR'}, inplace=True)

pval_v1.rename(columns={'V1': 'pval'}, inplace=True)
pval_v2.rename(columns={'V2': 'pval'}, inplace=True)
pval_v3.rename(columns={'V3': 'pval'}, inplace=True)
```

[ ]:
```
# Extraction des bornes inférieure et supérieure des CI
ci_v1[["ll", "hl"]] = ci_v1["CI"].str.strip("[]").str.split(",", expand=True).
 ↪astype(float)
ci_v2[["ll", "hl"]] = ci_v2["CI"].str.strip("[]").str.split(",", expand=True).
 ↪astype(float)
ci_v3[["ll", "hl"]] = ci_v3["CI"].str.strip("[]").str.split(",", expand=True).
 ↪astype(float)
```

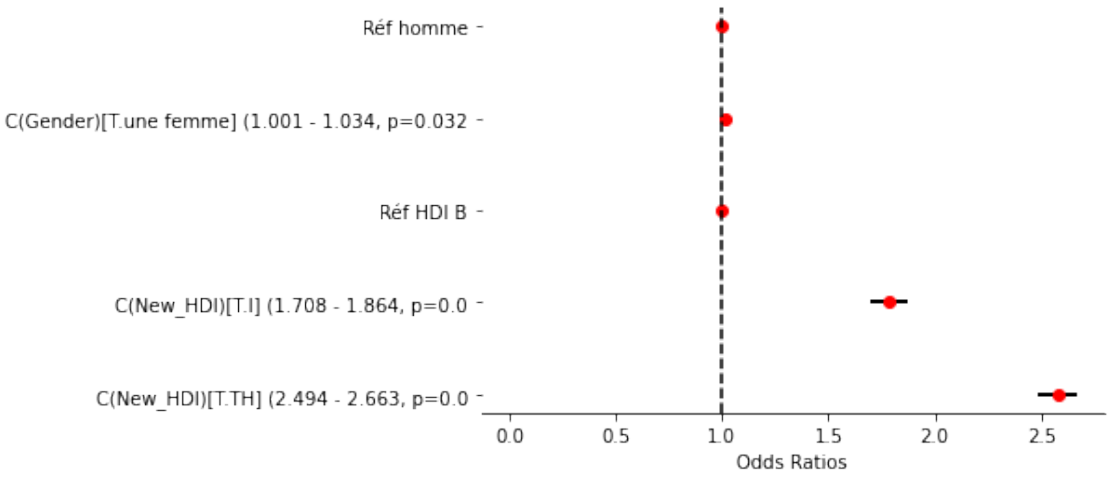
[ ]:
```
# Tableaux des valeurs pour créer le forestplot
forest_v1 = ci_v1.merge(or_v1, on="coef").merge(pval_v1, on="coef")
forest_v2 = ci_v2.merge(or_v2, on="coef").merge(pval_v2, on="coef")
forest_v3 = ci_v3.merge(or_v3, on="coef").merge(pval_v3, on="coef")
```

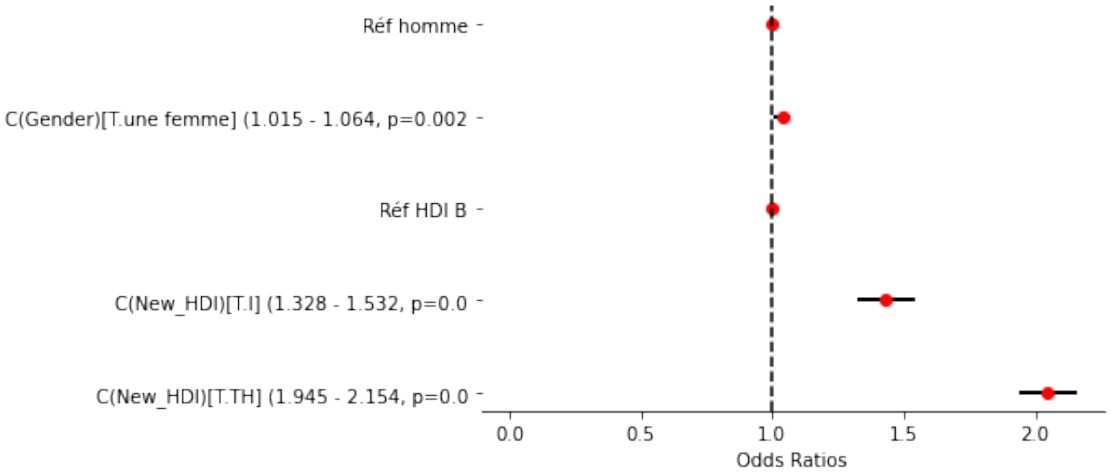
[ ]:
```
# Renommage de la référence "un homme"
forest_v1.iloc[0, 0] = "Réf homme"
forest_v2.iloc[0, 0] = "Réf homme"
forest_v3.iloc[0, 0] = "Réf homme"

# Ajout de la 2eme référence (HDI B)
full_pvalues = pd.DataFrame(full_pvalues)
forest_v1 = forest_v1.append(pd.Series(["Réf HDI B", "[0,0]", 0, 0, 1, ""],␣
 ↪index=forest_v1.columns), ignore_index=True)
forest_v2 = forest_v2.append(pd.Series(["Réf HDI B", "[0,0]", 0, 0, 1, ""],␣
 ↪index=forest_v2.columns), ignore_index=True)
forest_v3 = forest_v3.append(pd.Series(["Réf HDI B", "[0,0]", 0, 0, 1, ""],␣
 ↪index=forest_v3.columns), ignore_index=True)
```

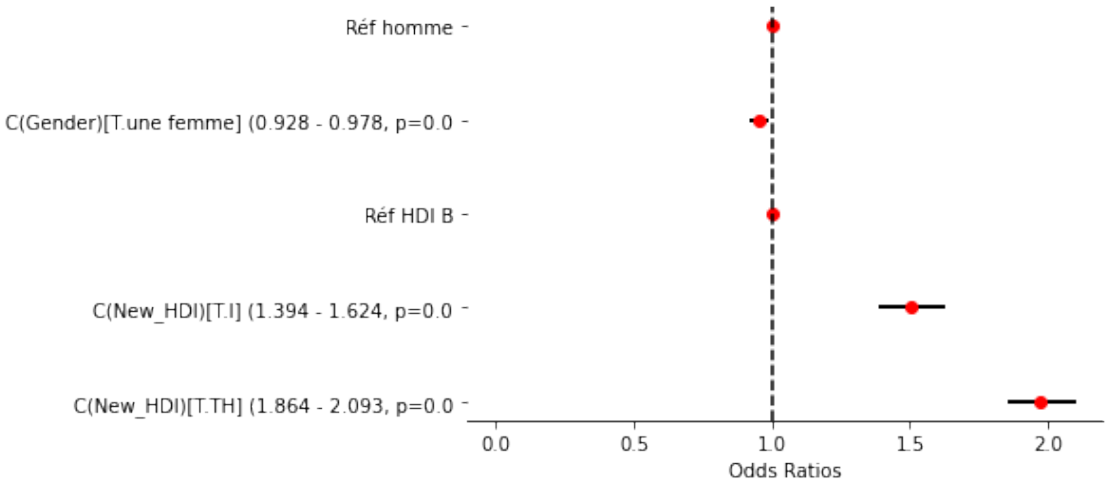
[ ]:
```
forest = []
for v in [forest_v1, forest_v2, forest_v3]:
    v.iloc[[2, 4]] = v.iloc[[4, 2]]
    v.iloc[[3, 4]] = v.iloc[[4, 3]]
    # Mise à zero des intervalles de confiance de l'intercept
    v.iloc[0, 2] = 0
    v.iloc[0, 3] = 0
    v.iloc[2, 2] = 0
```

```
    v.iloc[2, 3] = 0
    forest.append(v)
```

[ ]:
```
for n, version in enumerate(forest):
    # Données de la régression logistique
    version = version[::-1]
    odds_ratios = version["OR"].tolist()
    ci = list(zip(version["ll"], version["hl"]))
    ci_inf = version["ll"].astype(str)
    ci_sup = version["hl"].astype(str)
    labels = version["coef"] + " (" + ci_inf + " - " + ci_sup + ", p=" +␣
 ↪version["pval"].astype(str)
    labels.loc[0] = "Réf homme"
    labels.loc[2] = "Réf HDI B"

    # Création du forest plot
    fig, ax = plt.subplots()

    # OR, intervalles de confiance
    for i, (or_val, (ci_low, ci_high), label) in enumerate(zip(odds_ratios, ci,␣
 ↪labels)):
        ax.plot([ci_low, ci_high], [i, i], color="black", linewidth=2)
        ax.plot(or_val, i, 'ro')

    # Réglages des axes
    ax.set_xlim()
    ax.set_yticks(range(len(labels)))
    ax.set_yticklabels(labels)
    ax.set_xlabel('Odds Ratios')
    ax.axvline(1, color='black', linestyle='--')
    ax.spines["top"].set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["right"].set_visible(False)
    #plt.title()
    file_name = "../graph/forestplot_V" + str(n) + "_poisson.png"
    #plt.savefig(file_name)
    plt.show()
```


Réf homme
C(Gender)[T.une femme] (1.001 - 1.034, p=0.032
Réf HDI B
C(New_HDI)[T.I] (1.708 - 1.864, p=0.0
C(New_HDI)[T.TH] (2.494 - 2.663, p=0.0
0.0
0.5
1.0
1.5
2.0
2.5
Odds Ratios


Réf homme
C(Gender)[T.une femme] (1.015 - 1.064, p=0.002
Réf HDI B
C(New_HDI)[T.I] (1.328 - 1.532, p=0.0
C(New_HDI)[T.TH] (1.945 - 2.154, p=0.0
0.0
0.5
1.0
1.5
2.0
Odds Ratios


Réf homme
C(Gender)[T.une femme] (0.928 - 0.978, p=0.0
Réf HDI B
C(New_HDI)[T.I] (1.394 - 1.624, p=0.0
C(New_HDI)[T.TH] (1.864 - 2.093, p=0.0
0.0
0.5
1.0
1.5
2.0
Odds Ratios

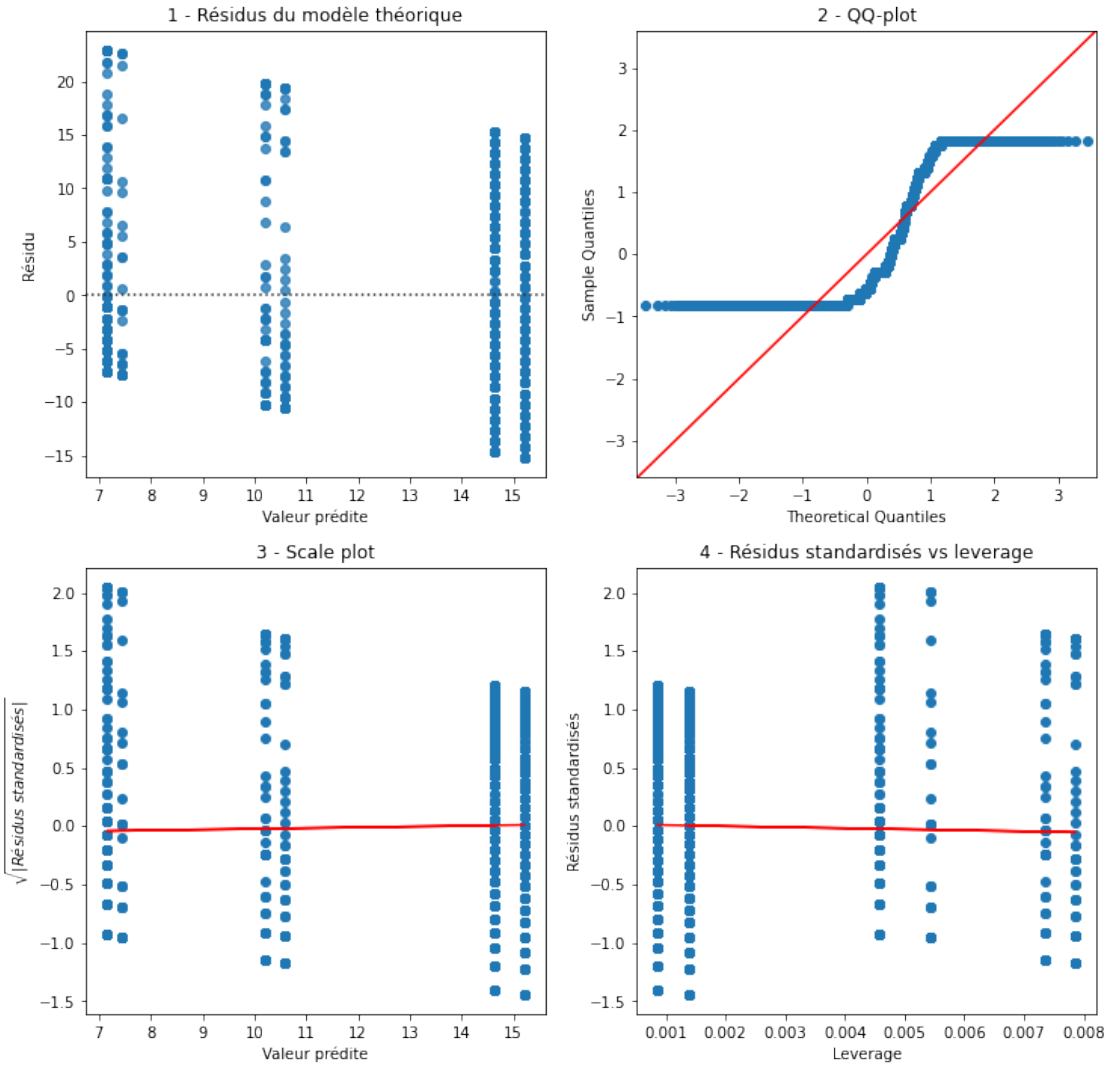
[ ]:
```
df_video_v2
```

[ ]:
```
               Gender New_HDI  video  predicted
Student_ID
32360       une femme      TH     24  15.206146
27808        un homme       B      4   7.151457
27532        un homme      TH     27  14.637709
2630         un homme       B     30   7.151457
23971        un homme      TH      6  14.637709
…                …     …     …         …
29275       une femme      TH     21  15.206146
28828        un homme       I      1  10.201600
26940        un homme      TH     23  14.637709
28699       une femme       I      2  10.597766
27897        un homme      TH      5  14.637709

[2154 rows x 4 columns]
```

[ ]:
```
# Calcul des résidus standardisés
residus = result_v2.resid_deviance
resid_std = (residus - np.mean(residus)) / np.std(residus)
# Calcul de la racine des résidus
residuals_sqrt = np.sqrt(np.abs(resid_std))
```

[ ]:
```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(12, 12))
# Graph homoscédasticité
sns.residplot(data=df_video_v2, x="predicted", y="video", ax=ax1)
ax1.set_title("1 - Résidus du modèle théorique")
ax1.set_ylabel("Résidu")
ax1.set_xlabel("Valeur prédite")
# Test de normalité de la distribution du nombre de videos (Q-Q plot) pour le␣
 ↪mooc version 2
qqplot(data=total_video.loc[2, "video"], fit=True, line="45", ax=ax2)
ax2.set_title("2 - QQ-plot")
# Racine carrée des résidus normalisés vs valeurs prédites
ax3.scatter(df_video_v2["predicted"], resid_std)
m, b = np.polyfit(df_video_v2["predicted"], resid_std, deg=1)
ax3.plot(df_video_v2["predicted"], m * df_video_v2["predicted"] + b,␣
 ↪color='red')
ax3.set_title("3 - Scale plot")
ax3.set_xlabel("Valeur prédite")
ax3.set_ylabel('$\sqrt{|Résidus\ standardisés|}$')
# residus standardisés vs leverage
influence = result_v2.get_influence()
```

```
leverage = influence.hat_matrix_diag
cooks_d = influence.cooks_distance[0]
ax4.scatter(leverage, resid_std)
m, b = np.polyfit(leverage, resid_std, deg=1)
ax4.plot(leverage, m * leverage + b, color='red')
# Afficher les points ayant un Cook's distance élevé
#threshold = 4 / len(leverage)  # Choisissez votre propre seuil
#high_cook_idx = np.where(cooks_d > threshold[0])
#ax4.scatter(leverage[high_cook_idx], resid_std[high_cook_idx], color='orange',␣
 ↪label='Cook\'s Distance > seuil')


ax4.set_xlabel('Leverage')
ax4.set_ylabel('Résidus standardisés')
ax4.set_title('4 - Résidus standardisés vs leverage')

plt.savefig("../graph/graphs_poisson.png")
plt.show()
```


1 - Résidus du modèle théorique
Résidu
Valeur prédite
2 - QQ-plot
Sample Quantiles
Theoretical Quantiles
3 - Scale plot
$\sqrt{|Résidus\ standardisés|}$
Valeur prédite
4 - Résidus standardisés vs leverage
Résidus standardisés
Leverage