Devoir final UE n°3 - Notebook Jupyter

Quentin Fouché

July 8, 2022

1 Préparation du jeu de données

1.1 Exploration des 7 jeux de données

[1]:
```
# (1) Import des packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
from statsmodels.formula.api import ols
import statsmodels.api as sm
from statsmodels.graphics.mosaicplot import mosaic
from statsmodels.api import qqplot
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from scipy import stats
from statsmodels.formula.api import logit
from statsmodels.formula.api import glm
from statsmodels.genmod import families
```

[2]:
```
# (2) Import du jeu de données "countries.HDI.csv"
countries_HDI = pd.read_csv(r"C:\Users\quent\Documents\DU Data Analyst␣
↪2022\UE n°3 - Introduction aux statistiques\Jeux de données\countries.HDI.
↪csv", low_memory = False, encoding = "latin-1")
first_line = list(countries_HDI.columns)
countries_HDI = countries_HDI.rename(columns = {"Norvège": "Country", "TH":␣
↪"Country_HDI", "1": "Country_rank"})
countries_HDI = countries_HDI.append({"Country": first_line[0], "Country_HDI":
↪ first_line[1], "Country_rank": first_line[2]}, ignore_index = True)
```

[3]:
```
# (3) Import des jeux de données sur les questionnaires
effec1_quest_compil = pd.read_csv(r"C:\Users\quent\Documents\DU Data Analyst␣
↪2022\UE n°3 - Introduction aux statistiques\Jeux de données\effec1.quest.
↪compil.csv", low_memory = False, encoding = "latin-1")
effec2_quest_compil = pd.read_csv(r"C:\Users\quent\Documents\DU Data Analyst␣
↪2022\UE n°3 - Introduction aux statistiques\Jeux de données\effec2.quest.
↪compil.csv", low_memory = False, encoding = "latin-1")
effec3_quest_compil = pd.read_csv(r"C:\Users\quent\Documents\DU Data Analyst␣
↪2022\UE n°3 - Introduction aux statistiques\Jeux de données\effec3.quest.
↪compil.csv", low_memory = False, encoding = "latin-1")
```

```
effec3_quest_compil = effec3_quest_compil[effec3_quest_compil["Student_ID"].
↪notnull()] # (sélection des lignes ayant des valeurs non-nulles pour␣
↪l'identifiant étudiant)
```

[4]:
```
# (4) Import des jeux de données de logs
usages_effec1 = pd.read_csv(r"C:\Users\quent\Documents\DU Data Analyst␣
↪2022\UE n°3 - Introduction aux statistiques\Jeux de données\usages.effec1.
↪csv", low_memory = False, encoding = "latin-1")
usages_effec2 = pd.read_csv(r"C:\Users\quent\Documents\DU Data Analyst␣
↪2022\UE n°3 - Introduction aux statistiques\Jeux de données\usages.effec2.
↪csv", low_memory = False, encoding = "latin-1")
usages_effec3 = pd.read_csv(r"C:\Users\quent\Documents\DU Data Analyst␣
↪2022\UE n°3 - Introduction aux statistiques\Jeux de données\usages.effec3.
↪csv", low_memory = False, encoding = "latin-1")
```

1.2 Fusion des 7 jeux de données pour obtenir une unique base de données rectangulaire

[5]:
```
# (1) Fusion des données de questionnaires et de logs pour chaque itération
iteration1 = pd.merge(effec1_quest_compil, usages_effec1, how = "outer", on =␣
↪"Student_ID")
iteration2 = pd.merge(effec2_quest_compil, usages_effec2, how = "outer", on =␣
↪"Student_ID")
iteration3 = pd.merge(effec3_quest_compil, usages_effec3, how = "outer", on =␣
↪"Student_ID")
```

[6]:
```
# (2) Ajout d'une colonne "iteration" dans les 3 tableaux créés
iteration1["Iteration"] = [1] * iteration1.shape[0]
iteration2["Iteration"] = [2] * iteration2.shape[0]
iteration3["Iteration"] = [3] * iteration3.shape[0]
```

[7]:
```
# (3) Fusion des trois itérations
iteration123 = pd.concat([iteration1, iteration2, iteration3])
```

[8]:
```
# (4) Ajout des données sur l'IDH des pays
countries_HDI = countries_HDI[countries_HDI["Country"].notnull()]
countries_HDI = countries_HDI.drop(columns = ["Country_HDI"])
df_final = pd.merge(iteration123, countries_HDI, how = "left", on = "Country")
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17411 entries, 0 to 17410
Columns: 121 entries, Student_ID to Country_rank
dtypes: float64(87), int64(1), object(33)
memory usage: 16.2+ MB
```

1.3 Simplification du jeu de données final

[9]:
```
# (1) Suppression de toutes les variables issues des questionnaires à␣
↪l'exception de Student_ID, Gender, Country, Country_HDI et Country_HDI.fin
columns_to_remove = list((list(effec1_quest_compil.columns),␣
↪list(effec2_quest_compil.columns), list(effec3_quest_compil.columns)))
columns_to_remove = list(itertools.chain.from_iterable(columns_to_remove))
columns_to_remove = list(dict.fromkeys(columns_to_remove))
for i in ["Student_ID", "Gender", "Country", "Country_HDI", "Country_HDI.
↪fin"]:
    columns_to_remove.remove(i)
df_final = df_final.drop(columns = columns_to_remove)
```

[10]:
```
# (2) Remplacement des caractères "Ã\x89" par "É"
df_final = df_final.replace("Ã\x89", "É", regex = True)
```

1.4 Création de trois nouvelles variables et calcul du nombre d'apprenants par catégorie d'HDI

[11]:
```
# (1) Calcul du nombre total de vidéos visionnées par apprenant ; les vidéos␣
↪de présentation (ex : "Prez.sem.1") sont exclues du calcul
my_list = list(df_final.iloc[:, 20:55].columns)
letter_S = "S"
columns_videos = [idx for idx in my_list if idx[0].lower() == letter_S.
↪lower()]
df_final_videos = df_final.loc[:,columns_videos]
df_final["Nb_tot_video"] = df_final_videos.sum(axis = 1, skipna = False)
df_final["Nb_tot_video"].describe()
```

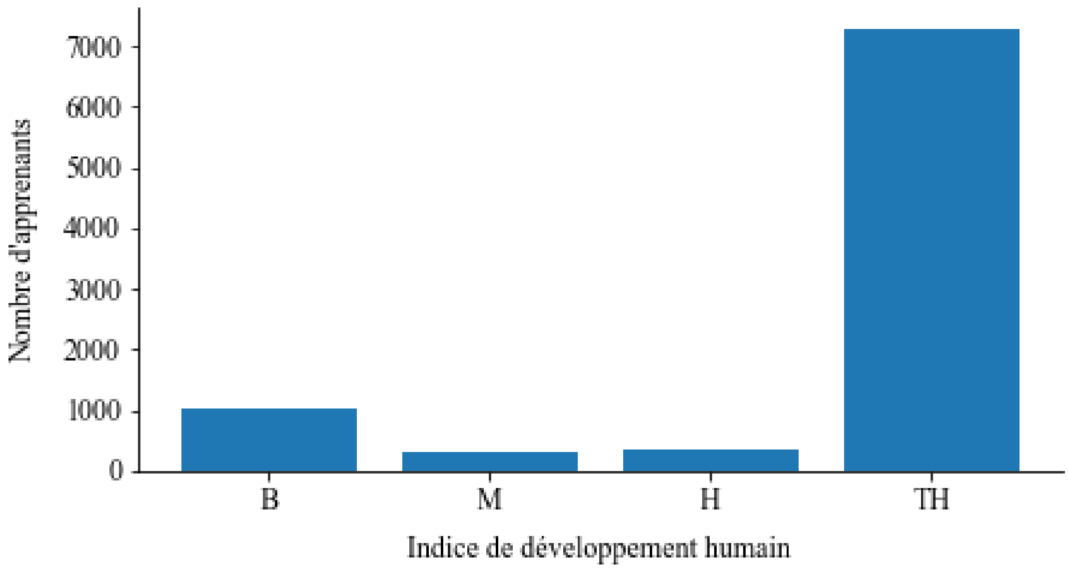
[11]:
```
count    15646.000000
mean         8.909498
std         11.236576
min          0.000000
25%          0.000000
50%          2.000000
75%         17.000000
max         30.000000
Name: Nb_tot_video, dtype: float64
```

[12]:
```
# (2) Calcul du nombre total de quiz réalisés par apprenant
df_final_quiz = df_final.iloc[:, [10, 12, 14, 15 ,17]]
df_final["Nb_tot_quiz"] = df_final_quiz.sum(axis = 1, skipna = False)
df_final["Nb_tot_quiz"].describe()
```

[12]:
```
count    15646.000000
mean         1.932890
std          2.185275
min          0.000000
25%          0.000000
50%          1.000000
75%          5.000000
```

```
max          5.000000
Name: Nb_tot_quiz, dtype: float64
```

[13]:
```
# (3) Calcul du nombre d'apprenants par catégorie d'IDH et représentation␣
↪graphique
Nb_cat_HDI = df_final.groupby("Country_HDI").size()
plt.rc("font", family = "Times New Roman", size = 12)
fig, ax = plt.subplots(figsize = (17.5/2.54, 9/2.54))
ax.bar(x = [Nb_cat_HDI.index[0], Nb_cat_HDI.index[2], Nb_cat_HDI.index[1],␣
↪Nb_cat_HDI.index[3]], height = [Nb_cat_HDI.values[0], Nb_cat_HDI.values[2],␣
↪Nb_cat_HDI.values[1], Nb_cat_HDI.values[3]])
ax.spines[["right", "top"]].set_visible(False)
ax.set_xlabel("Indice de développement humain", labelpad = 8)
ax.set_ylabel("Nombre d'apprenants", labelpad = 10)
plt.show()
```



[14]:
```
# (4) Recodage de l'IDH en regroupant les catégories "M" et "H" en une␣
↪catégorie intermédiaire "I"
df_final["Country_HDI_rec"] = df_final["Country_HDI"]
df_final["Country_HDI_rec"] = df_final["Country_HDI_rec"].str.replace("M",␣
↪"I")
df_final["Country_HDI_rec"] = df_final["Country_HDI_rec"].str.replace("H",␣
↪"I")
df_final["Country_HDI_rec"] = df_final["Country_HDI_rec"].str.replace("TI",␣
↪"TH")
df_final.groupby("Country_HDI_rec").size()
```

[14]:
```
Country_HDI_rec
B      1032
I       667
TH     7270
```

```
dtype: int64
```

[15]:
```
# (5) Nouvelle simplification du jeu de données final et description des␣
↪variables
df_final = df_final[["Student_ID", "Gender", "Country_HDI_rec", "Exam.bin",␣
↪"Assignment.bin", "Quizz.1.bin", "Quizz.2.bin", "Quizz.3.bin", "Quizz.4.
↪bin", "Quizz.5.bin", "Nb_tot_video", "Nb_tot_quiz", "Iteration"]]
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17411 entries, 0 to 17410
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Student_ID       17411 non-null  float64
 1   Gender           9099 non-null   object
 2   Country_HDI_rec  8969 non-null   object
 3   Exam.bin         15646 non-null  float64
 4   Assignment.bin   15646 non-null  float64
 5   Quizz.1.bin      15646 non-null  float64
 6   Quizz.2.bin      15646 non-null  float64
 7   Quizz.3.bin      15646 non-null  float64
 8   Quizz.4.bin      15646 non-null  float64
 9   Quizz.5.bin      15646 non-null  float64
 10  Nb_tot_video     15646 non-null  float64
 11  Nb_tot_quiz      15646 non-null  float64
 12  Iteration        17411 non-null  int64
dtypes: float64(10), int64(1), object(2)
memory usage: 1.9+ MB
```

2 Description du jeu de données

2.1 Définition des quatre types d'apprenants (Bystander, Auditing, Completer et Disengaging)

[16]:
```
# (1) Définition des apprenants "Completer"
Completer = []
for i in range(0, len(df_final.index)):
    if df_final.loc[i, "Exam.bin"] == 1:
        Completer.append("Completer")
    else:
        Completer.append("Other")
df_final["Student_type"] = Completer
```

[17]:
```
# (2) Définition des apprenants "Disengaging"
Disengaging = []
for i in df_final.index:
    if df_final.loc[i, "Student_type"] == "Other":
        if True in (df_final.loc[i, ["Assignment.bin", "Quizz.1.bin", "Quizz.
↪2.bin", "Quizz.3.bin", "Quizz.4.bin" ,"Quizz.5.bin"]] == 1).values:
            Disengaging.append("Disengaging")
```

```
        else:
            Disengaging.append("Other")
    else:
        Disengaging.append(df_final.loc[i, "Student_type"])
df_final["Student_type"] = Disengaging
```

[18]:
```
# (3) Définition des apprenants "Auditing"
Auditing = []
for i in df_final.index:
    if df_final.loc[i, "Student_type"] == "Other":
        if df_final.loc[i, "Nb_tot_video"] >= 6:
            Auditing.append("Auditing")
        else:
            Auditing.append("Other")
    else:
        Auditing.append(df_final.loc[i, "Student_type"])
df_final["Student_type"] = Auditing
```

[19]:
```
# (4) Définition des apprenants "Bystander" (et attribution de la valeur␣
↪"Unknown" pour tous les apprenants restants non-classés)
Bystander = []
for i in df_final.index:
    if df_final.loc[i, "Student_type"] == "Other":
        if df_final.loc[i, "Nb_tot_video"] < 6:
            Bystander.append("Bystander")
        else:
            Bystander.append("Unknown")
    else:
        Bystander.append(df_final.loc[i, "Student_type"])
df_final["Student_type"] = Bystander
df_final.groupby("Student_type").size()
```

[19]:
```
Student_type
Auditing        365
Bystander      6839
Completer      1741
Disengaging    6701
Unknown        1765
dtype: int64
```

2.2 Calcul de la proportion des quatres types d'apprenants en fonction de l'itération du MOOC

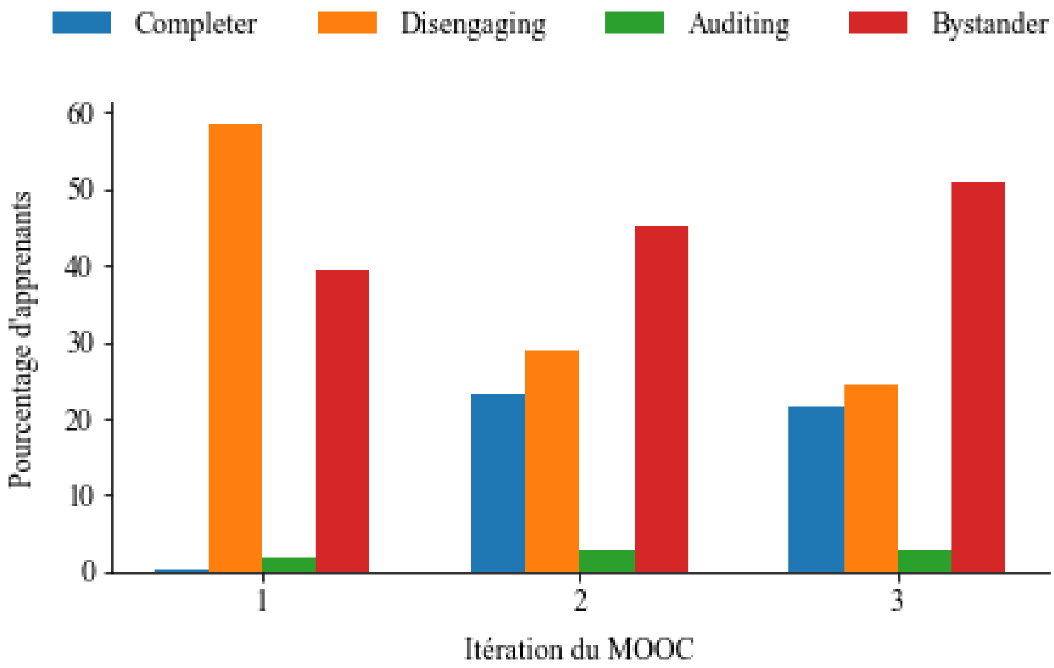
[20]:
```
# (1) Calcul du nombre d'apprenants par type et par itération du MOOC
df_final_without_unknown = df_final[df_final["Student_type"] != "Unknown"]
Nb_type_iteration = df_final_without_unknown.pivot_table(values =␣
↪"Student_ID", index = "Student_type", columns = "Iteration", aggfunc =␣
↪"count")
Nb_type_iteration
```

[20]:
```
Iteration        1     2     3
Student_type
Auditing       152   106   107
Bystander     3139  1720  1980
Completer       20   878   843
Disengaging   4654  1094   953
```

[21]:
```
# (2) Conversion en pourcentage
Prop_type_iteration = Nb_type_iteration.copy(deep = True)
for i in range(1,4):
    Prop_type_iteration[i] = (Prop_type_iteration[i] / Prop_type_iteration[i].
 ↪sum()) * 100
Prop_type_iteration
```

[21]:
```
Iteration             1          2          3
Student_type
Auditing       1.908349   2.790943   2.755601
Bystander     39.409918  45.286993  50.991501
Completer      0.251099  23.117430  21.710018
Disengaging   58.430634  28.804634  24.542879
```

[22]:
```
# (3) Représentation du nombre d'apprenants en fonction de leur type et de␣
 ↪l'itération du MOOC
plt.rc("font", family = "Times New Roman", size = 12)
fig, ax = plt.subplots(figsize = (17.5/2.54, 9/2.54))
labels = ["1", "2", "3"]
x = np.arange(len(labels))
width = 0.17
ax.bar(x - 1.5 * width, [Prop_type_iteration.loc["Completer", 1],␣
 ↪Prop_type_iteration.loc["Completer", 2], Prop_type_iteration.
 ↪loc["Completer", 3]], width, label = "Completer")
ax.bar(x - width/2, [Prop_type_iteration.loc["Disengaging", 1],␣
 ↪Prop_type_iteration.loc["Disengaging", 2], Prop_type_iteration.
 ↪loc["Disengaging", 3]], width, label = "Disengaging")
ax.bar(x + width/2, [Prop_type_iteration.loc["Auditing", 1],␣
 ↪Prop_type_iteration.loc["Auditing", 2], Prop_type_iteration.loc["Auditing",␣
 ↪3]], width, label = "Auditing")
ax.bar(x + 1.5 * width, [Prop_type_iteration.loc["Bystander", 1],␣
 ↪Prop_type_iteration.loc["Bystander", 2], Prop_type_iteration.
 ↪loc["Bystander", 3]], width, label = "Bystander")
ax.spines[["right", "top"]].set_visible(False)
ax.set_xticks(x, labels)
ax.set_xlabel("Itération du MOOC", labelpad = 8)
ax.set_ylabel("Pourcentage d'apprenants", labelpad = 10)
ax.legend(bbox_to_anchor=(1.03, 1.25), frameon = False, ncol = 4)
plt.show()
```



3 Chi2 et mosaic plot

[23]:
```
# (1) Calcul du nombre d'apprenants par catégorie d'IDH en fonction du genre
Nb_HDI_gender = df_final.pivot_table(values = "Student_ID", index = "Gender",␣
↪columns = "Country_HDI_rec", aggfunc = "count")
Nb_HDI_gender
```

[23]:
```
Country_HDI_rec    B    I    TH
Gender
un homme         883  432  4716
une femme        147  233  2546
```

[24]:
```
# (2) Test du chi2 d'indépendance
stats.chi2_contingency(Nb_HDI_gender)
```

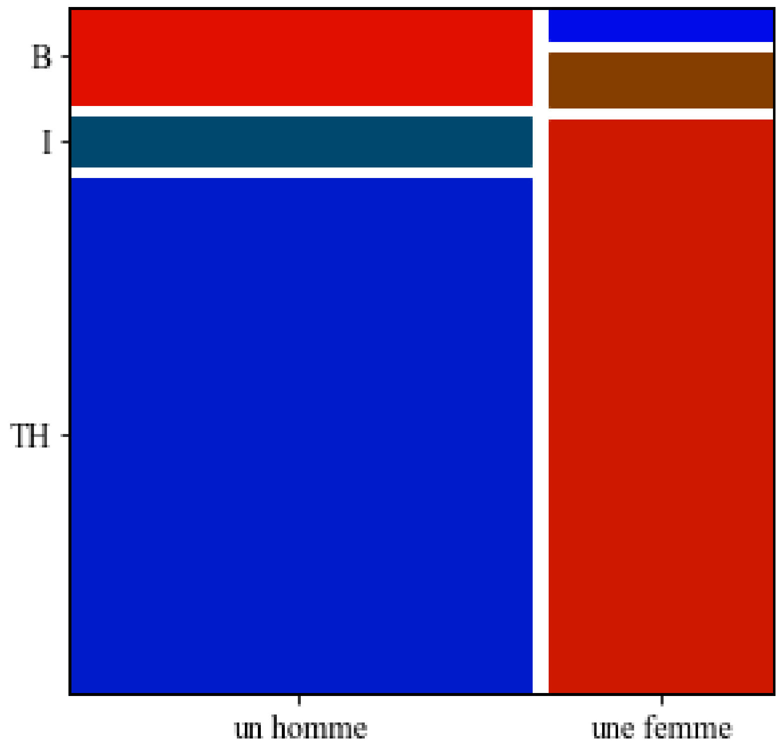
[24]:
```
(179.0476122707751,
 1.3191828493466097e-39,
 2,
 array([[ 693.52796695,  447.76320196, 4889.70883108],
        [ 336.47203305,  217.23679804, 2372.29116892]]))
```

[25]:
```
# (3) Affichage des résidus
table = sm.stats.Table(Nb_HDI_gender)
table.resid_pearson
```

[25]:
```
Country_HDI_rec          B         I        TH
Gender
un homme          7.194707 -0.744938 -2.484165
```

```
une femme       -10.329294  1.069493  3.566464
```

[26]:
```
# (4) Représentation des résidus en mosaic plot (résidus positifs en rouge,␣
↪négatifs en bleu)
data = {("un homme", "TH"): 4716, ("un homme", "I"): 432, ("un homme", "B"):␣
↪883,
        ("une femme", "TH"): 2546, ("une femme", "I"): 233, ("une femme",␣
↪"B"): 147}
labelizer = lambda k: {("un homme", "TH"): "", ("un homme", "I"): "", ("un␣
↪homme", "B"): "",
                       ("une femme", "TH"): "", ("une femme", "I"): "", ("une␣
↪femme", "B"): ""}[k]
plt.rc("font", family = "Times New Roman", size = 12)
plt.rcParams["figure.figsize"] = [12/2.54, 12/2.54]
mosaic(data, gap = 0.03, labelizer = labelizer, statistic = True)
plt.show()
```



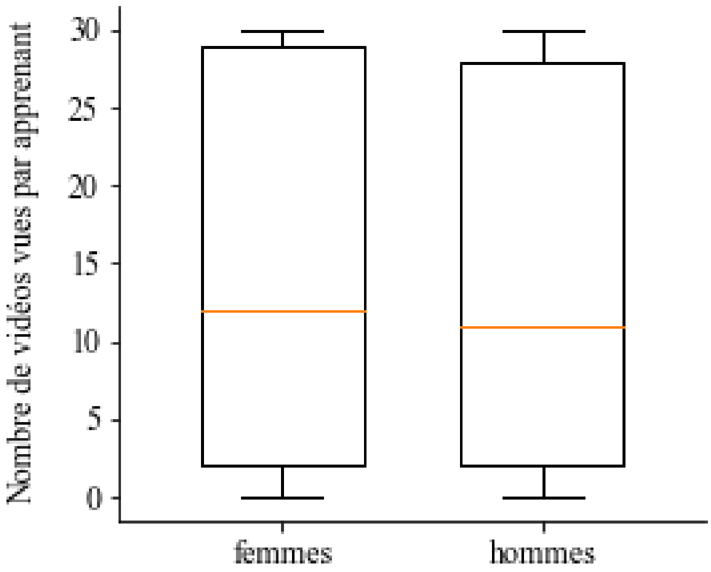
4 Modèle linéaire, tests non paramétriques

4.1 Test t de Student et test de Mann-Whitney

[27]:
```
# (1) Regroupement du nombre de vidéos visionnées par apprenant en fonction␣
↪du genre
Nb_tot_video_femmes = df_final[df_final["Gender"] == "une␣
↪femme"]["Nb_tot_video"].dropna()
```

```
Nb_tot_video_hommes = df_final[df_final["Gender"] == "un␣
↪homme"]["Nb_tot_video"].dropna()
```

[28]:
```
# (2) Représentation par boxplot de la distribution du nombre de vidéos vues␣
↪en fonction du genre
plt.rc("font", family = "Times New Roman", size = 12)
fig, ax = plt.subplots(figsize = (10/2.54, 9/2.54))
ax.boxplot([Nb_tot_video_femmes, Nb_tot_video_hommes], positions = [1, 1.8],␣
↪widths = 0.5, labels = ["femmes", "hommes"])
ax.spines[["right", "top"]].set_visible(False)
ax.set_ylabel("Nombre de vidéos vues par apprenant", labelpad = 10)
plt.show()
```



[29]:
```
# (3) Test t de Student pour comparer le nombre de vidéos vues entre les␣
↪genres
stats.ttest_ind(Nb_tot_video_femmes, Nb_tot_video_hommes, equal_var = True,␣
↪nan_policy = "omit")
```

[29]:
```
Ttest_indResult(statistic=3.7828059599346706, pvalue=0.00015606501583036533)
```

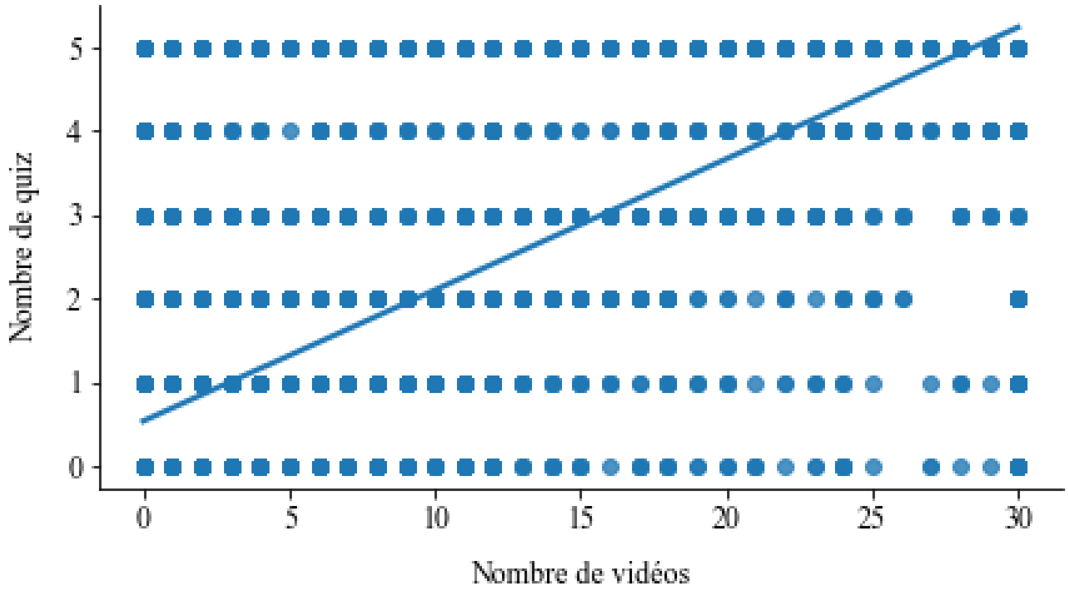
[30]:
```
# (4) Test de Mann-Whitney pour la même comparaison
stats.mannwhitneyu(Nb_tot_video_femmes, Nb_tot_video_hommes)
```

[30]:
```
MannwhitneyuResult(statistic=9536047.0, pvalue=0.00043347156469210563)
```

4.2 Régression linéaire

[31]:
```
# (1) Représentation du nombre de vidéos visionnées par apprenant en fonction␣
↪du nombre de quiz réalisés (scatterplot)
plt.rc("font", family = "Times New Roman", size = 12)
plt.figure(figsize = (17.5/2.54, 9/2.54))
```

```python
sns.regplot(x = "Nb_tot_video", y = "Nb_tot_quiz", data = df_final, ci = None)
plt.ylabel("Nombre de quiz", labelpad = 10)
plt.xlabel("Nombre de vidéos", labelpad = 10)
plt.gca().spines[["right", "top"]].set_visible(False)
plt.show()
```



[32]:

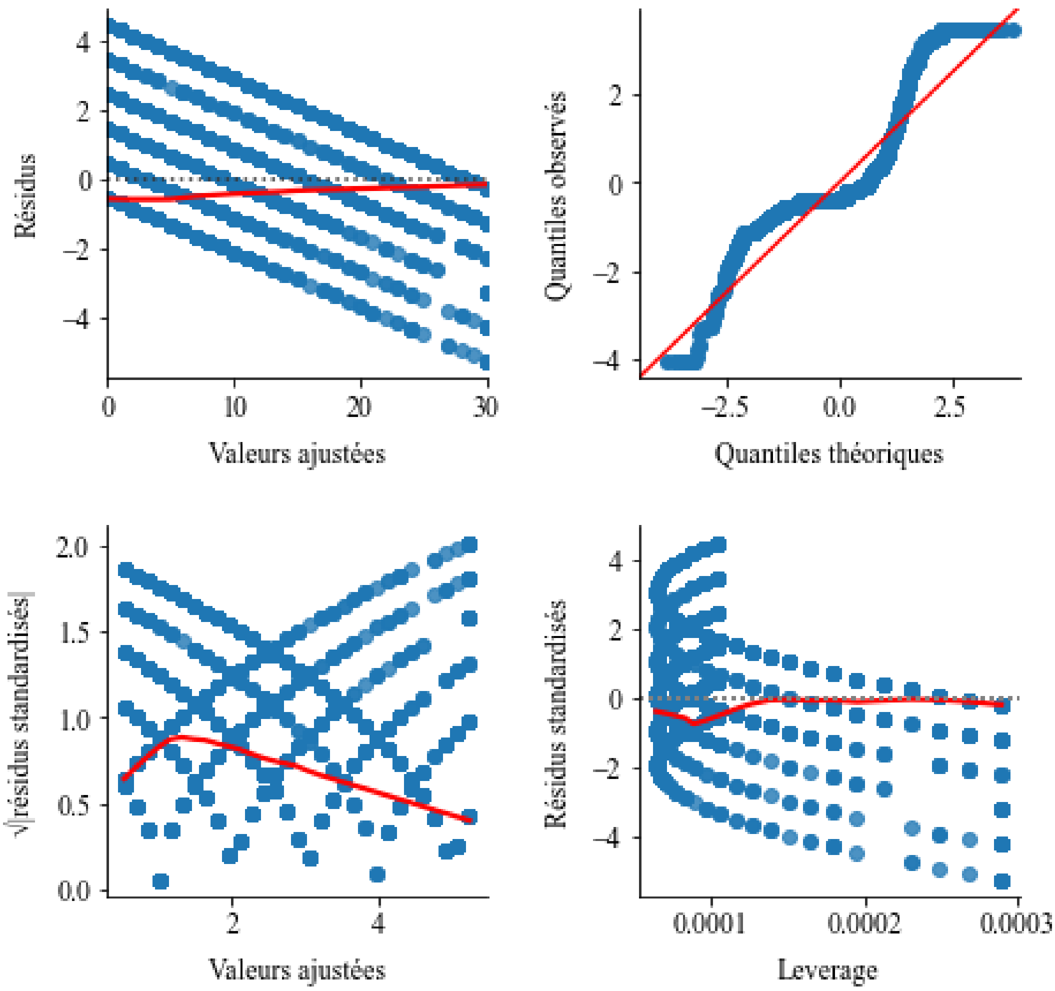
```python
# (2) Création d'un modèle linéaire et affichage des paramètres
my_model = ols("Nb_tot_quiz ~ Nb_tot_video", data = df_final).fit()
print(my_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Nb_tot_quiz   R-squared:                       0.652
Model:                            OLS   Adj. R-squared:                  0.652
Method:                 Least Squares   F-statistic:                 2.930e+04
Date:                Fri, 08 Jul 2022   Prob (F-statistic):               0.00
Time:                        19:29:27   Log-Likelihood:                -26175.
No. Observations:               15646   AIC:                         5.235e+04
Df Residuals:                   15644   BIC:                         5.237e+04
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept        0.5338      0.013     40.583      0.000       0.508       0.560
Nb_tot_video     0.1570      0.001    171.184      0.000       0.155       0.159
==============================================================================
Omnibus:                     5312.751   Durbin-Watson:                   1.892
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            17495.515
Skew:                           1.743   Prob(JB):                         0.00
Kurtosis:                       6.832   Cond. No.                         18.3
==============================================================================
```

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

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

[33]:

```python
# (3) Visualisation de l'adéquation du modèle
model_norm_residuals = my_model.get_influence().resid_studentized_internal
model_norm_residuals_abs_sqrt = np.sqrt(np.abs(model_norm_residuals))
summary_info = my_model.get_influence().summary_frame()
leverage = summary_info["hat_diag"]

plt.rc("font", family = "Times New Roman", size = 12)
fig, ax = plt.subplots(nrows = 2, ncols = 2, figsize = (18/2.54, 18/2.54))

sns.residplot(x = "Nb_tot_video", y = "Nb_tot_quiz", data = df_final, lowess␣
 ↪= True, line_kws = {"color": "red"}, ax = ax[0, 0])
ax[0, 0].set_xlabel("Valeurs ajustées", labelpad = 8)
ax[0, 0].set_ylabel("Résidus", labelpad = 8)
ax[0, 0].spines[["top", "right"]].set_visible(False)

qqplot(data = my_model.resid, fit = True, line = "45", ax = ax[0, 1])
ax[0, 1].set_xlabel("Quantiles théoriques", labelpad = 8)
ax[0, 1].set_ylabel("Quantiles observés", labelpad = 8)
ax[0, 1].spines[["top", "right"]].set_visible(False)

sns.regplot(x = my_model.fittedvalues, y = model_norm_residuals_abs_sqrt, ci␣
 ↪= None, lowess = True, line_kws = {"color": "red"}, ax = ax[1, 0])
ax[1, 0].set_xlabel("Valeurs ajustées", labelpad = 8)
ax[1, 0].set_ylabel("|résidus standardisés|", labelpad = 8)
ax[1, 0].spines[["top", "right"]].set_visible(False)

sns.regplot(x = leverage, y = my_model.resid, ci = None, lowess = True,␣
 ↪line_kws = {"color": "red"}, ax = ax[1, 1])
ax[1, 1].set_xlabel("Leverage", labelpad = 8)
ax[1, 1].set_ylabel("Résidus standardisés", labelpad = 8)
ax[1, 1].spines[["top", "right"]].set_visible(False)
ax[1, 1].axhline(y = 0, linewidth = 2, linestyle = (0, (1, 1)), color =␣
 ↪"grey")

plt.subplots_adjust(wspace = 0.4, hspace = 0.4)
```



[34]:
```
# (4) Test de corrélation de Pearson
stats.pearsonr(df_final["Nb_tot_video"].dropna(), df_final["Nb_tot_quiz"].
 ↪dropna())
```

[34]:
```
(0.8074356737743954, 0.0)
```

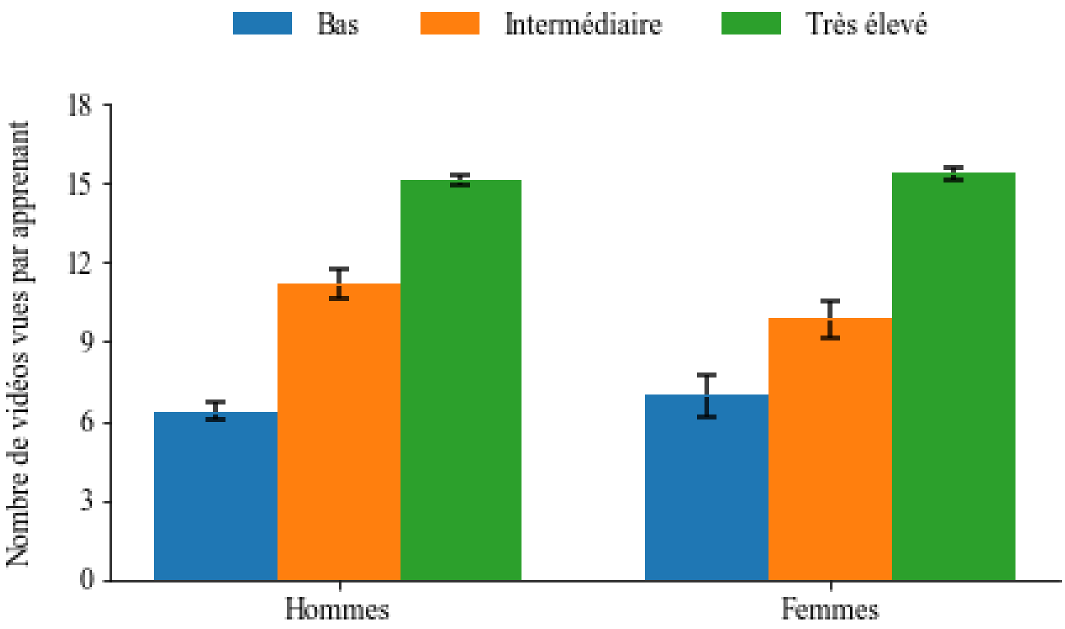
[35]:
```
# (5) Test de corrélation de Spearman
stats.spearmanr(df_final["Nb_tot_video"].dropna(), df_final["Nb_tot_quiz"].
 ↪dropna())
```

[35]:
```
SpearmanrResult(correlation=0.7997427789891928, pvalue=0.0)
```

4.3 ANOVA

[36]:
```
# (1) Représentation du nombre moyen de vidéos visionnées par apprenant en␣
 ↪fonction du genre et de l'IDH
Nb_tot_video_HDI_gender = df_final.pivot_table(values = "Nb_tot_video", index␣
 ↪= "Gender", columns = "Country_HDI_rec", aggfunc = "mean")
Nb_tot_video_HDI_gender_esm = df_final.pivot_table(values = "Nb_tot_video",␣
 ↪index = "Gender", columns = "Country_HDI_rec", aggfunc = "sem")
```

```python
plt.rc("font", family = "Times New Roman", size = 12)
fig, ax = plt.subplots(figsize = (17.5/2.54, 9/2.54))
labels = ["Hommes", "Femmes"]
x = np.arange(len(labels))
width = 0.25
ax.bar(x - width, [Nb_tot_video_HDI_gender.loc["un homme", "B"], Nb_tot_video_HDI_gender.loc["une femme", "B"]], width, label = "Bas")
ax.bar(x, [Nb_tot_video_HDI_gender.loc["un homme", "I"], Nb_tot_video_HDI_gender.loc["une femme", "I"]], width, label = "Intermédiaire")
ax.bar(x + width, [Nb_tot_video_HDI_gender.loc["un homme", "TH"], Nb_tot_video_HDI_gender.loc["une femme", "TH"]], width, label = "Très élevé")
ax.errorbar(x - width, [Nb_tot_video_HDI_gender.loc["un homme", "B"], Nb_tot_video_HDI_gender.loc["une femme", "B"]], yerr = [Nb_tot_video_HDI_gender_esm.loc["un homme", "B"], Nb_tot_video_HDI_gender_esm.loc["une femme", "B"]], fmt = "_", capsize = 4, capthick = 1.5, ecolor = "black")
ax.errorbar(x, [Nb_tot_video_HDI_gender.loc["un homme", "I"], Nb_tot_video_HDI_gender.loc["une femme", "I"]], yerr = [Nb_tot_video_HDI_gender_esm.loc["un homme", "I"], Nb_tot_video_HDI_gender_esm.loc["une femme", "I"]], fmt = "_", capsize = 4, capthick = 1.5, ecolor = "black")
ax.errorbar(x + width, [Nb_tot_video_HDI_gender.loc["un homme", "TH"], Nb_tot_video_HDI_gender.loc["une femme", "TH"]], yerr = [Nb_tot_video_HDI_gender_esm.loc["un homme", "TH"], Nb_tot_video_HDI_gender_esm.loc["une femme", "TH"]], fmt = "_", capsize = 4, capthick = 1.5, ecolor = "black")
ax.spines[["right", "top"]].set_visible(False)
ax.set_xticks(x, labels)
ax.set_ylabel("Nombre de vidéos vues par apprenant", labelpad = 10)
ax.set_yticks([0, 3, 6, 9, 12, 15, 18], ["0", "3", "6", "9", "12", "15", "18"])
ax.legend(bbox_to_anchor=(0.89, 1.25), frameon = False, ncol = 3)
plt.show()
```



[37]:
```
# (2) ANOVA à deux facteur sur le nombre de vidéos visionnées par apprenant␣
↪en fonction de l'IDH et du genre (sur les données des 3 itérations) ;␣
↪interaction non prise en compte
my_model1 = ols("Nb_tot_video ~ C(Country_HDI_rec) + C(Gender)", data =␣
↪df_final).fit()
print("Table d'ANOVA :")
print(sm.stats.anova_lm(my_model1, typ = 2))
print("")
print("Résumé du modèle avec les statistiques inférentielles :")
print(my_model1.summary())
print("")
print("Distribution des résidus :")
print(my_model1.resid.describe())
```

```
Table d'ANOVA :
                          sum_sq      df           F         PR(>F)
C(Country_HDI_rec)  7.415641e+04     2.0  284.661456  1.403405e-120
C(Gender)           5.743061e+01     1.0    0.440913   5.066990e-01
Residual            1.165380e+06  8947.0         NaN            NaN

Résumé du modèle avec les statistiques inférentielles :
                            OLS Regression Results
==============================================================================
Dep. Variable:           Nb_tot_video   R-squared:                       0.061
Model:                            OLS   Adj. R-squared:                  0.061
Method:                 Least Squares   F-statistic:                     194.6
Date:                Fri, 08 Jul 2022   Prob (F-statistic):          3.06e-122
Time:                        19:29:53   Log-Likelihood:                -34492.
No. Observations:                8951   AIC:                         6.899e+04
Df Residuals:                    8947   BIC:                         6.902e+04
```

```
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
============
                              coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------------
Intercept                   6.4327      0.358     17.991      0.000       5.732
7.134
C(Country_HDI_rec)[T.I]     4.2482      0.570      7.449      0.000       3.130
5.366
C(Country_HDI_rec)[T.TH]    8.7085      0.384     22.688      0.000       7.956
9.461
C(Gender)[T.une femme]      0.1725      0.260      0.664      0.507      -0.337
0.682
==============================================================================
Omnibus:                64231.085   Durbin-Watson:                   1.813
Prob(Omnibus):              0.000   Jarque-Bera (JB):              835.545
Skew:                       0.182   Prob(JB):                    3.66e-182
Kurtosis:                   1.548   Cond. No.                         7.78
==============================================================================

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

Distribution des résidus :
count    8.951000e+03
mean     2.110027e-14
std      1.141096e+01
min     -1.531363e+01
25%     -1.014112e+01
50%     -3.141118e+00
75%      1.368637e+01
max      2.356734e+01
dtype: float64
```

[38]:
```
# (3) Reprise de l'ANOVA précédente en prenant en compte l'interaction entre␣
↪genre et IDH
my_model2 = ols("Nb_tot_video ~ C(Country_HDI_rec) * C(Gender)", data =␣
↪df_final).fit()
print("Table d'ANOVA :")
print(sm.stats.anova_lm(my_model2, typ = 2))
print("")
print("Résumé du modèle avec les statistiques inférentielles :")
print(my_model2.summary())
print("")
print("Distribution des résidus :")
print(my_model2.resid.describe())
```

```
Table d'ANOVA :
```

```
                                    sum_sq      df           F         PR(>F)
C(Country_HDI_rec)            7.415641e+04     2.0  284.693011  1.364923e-120
C(Gender)                     5.743061e+01     1.0    0.440962   5.066754e-01
C(Country_HDI_rec):C(Gender)  3.896517e+02     2.0    1.495907   2.241013e-01
Residual                      1.164990e+06  8945.0         NaN            NaN

Résumé du modèle avec les statistiques inférentielles :
                            OLS Regression Results
==============================================================================
Dep. Variable:            Nb_tot_video   R-squared:                       0.062
Model:                             OLS   Adj. R-squared:                  0.061
Method:                  Least Squares   F-statistic:                     117.4
Date:                 Fri, 08 Jul 2022   Prob (F-statistic):         1.35e-120
Time:                         19:29:53   Log-Likelihood:                -34491.
No. Observations:                 8951   AIC:                         6.899e+04
Df Residuals:                     8945   BIC:                         6.904e+04
Df Model:                            5
Covariance Type:             nonrobust
================================================================================
===================================
                                                     coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
-----------------------------------
Intercept                                          6.3737      0.384     16.596
0.000       5.621       7.127
C(Country_HDI_rec)[T.I]                            4.8392      0.670      7.222
0.000       3.526       6.153
C(Country_HDI_rec)[T.TH]                           8.7296      0.418     20.859
0.000       7.909       9.550
C(Gender)[T.une femme]                             0.5855      1.017      0.576
0.565      -1.407       2.578
C(Country_HDI_rec)[T.I]:C(Gender)[T.une femme]    -1.9315      1.376     -1.403
0.161      -4.629       0.766
C(Country_HDI_rec)[T.TH]:C(Gender)[T.une femme]   -0.3053      1.055     -0.290
0.772      -2.373       1.762
==============================================================================
Omnibus:                     64400.643   Durbin-Watson:                   1.814
Prob(Omnibus):                   0.000   Jarque-Bera (JB):              834.435
Skew:                            0.181   Prob(JB):                    6.38e-182
Kurtosis:                        1.549   Cond. No.                         22.1
==============================================================================

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

Distribution des résidus :
count    8.951000e+03
mean     1.850777e-14
std      1.140905e+01
```

```
min     -1.538350e+01
25%     -1.010338e+01
50%     -3.103375e+00
75%      1.361650e+01
max      2.362627e+01
dtype: float64
```

[39]:
```python
# (4) Tests post-hoc de Tukey à partir de l'ANOVA précédente (en ne se␣
↪focalisant que sur les comparaisons entre deux catégories d'IDH au sein␣
↪d'un même genre)
anova_df = df_final.copy(deep = True)
anova_df = anova_df[anova_df["Nb_tot_video"].notnull()]
anova_df["Combination"] = anova_df.Country_HDI_rec + " / " + anova_df.Gender
anova_df = anova_df[anova_df["Combination"].notnull()]
m_comp = pairwise_tukeyhsd(endog = anova_df["Nb_tot_video"], groups =␣
↪anova_df["Combination"], alpha = 0.05)
tukey_data = pd.DataFrame(data = m_comp._results_table.data[1:], columns =␣
↪m_comp._results_table.data[0])
tukey_data.iloc[[1,3,6,8,10,13],:]
```

[39]:

	group1	group2	meandiff	p-adj	lower	upper	reject
1	B / un homme	I / un homme	4.8392	0.0000	2.9294	6.7491	True
3	B / un homme	TH / un homme	8.7296	0.0000	7.5368	9.9225	True
6	B / une femme	I / une femme	2.9078	0.1497	-0.5185	6.3340	False
8	B / une femme	TH / une femme	8.4243	0.0000	5.6650	11.1836	True
10	I / un homme	TH / un homme	3.8904	0.0000	2.2552	5.5256	True
13	I / une femme	TH / une femme	5.5165	0.0000	3.2901	7.7430	True

5 Régression logistique

5.1 Présenter des odd-ratios

[40]:
```python
# (1) Régression logistique multiple sur la réalisation de l'examen final en␣
↪fonction de l'IDH et du genre (sur les données des 3 itérations)
df_final["Exam_bin"] = df_final["Exam.bin"]
my_model = logit("Exam_bin ~ C(Country_HDI_rec) + C(Gender)", data =␣
↪df_final).fit()
print(my_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.476930
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:               Exam_bin   No. Observations:                 8951
Model:                          Logit   Df Residuals:                     8947
Method:                           MLE   Df Model:                            3
Date:                Fri, 08 Jul 2022   Pseudo R-squ.:                0.002390
Time:                        19:29:53   Log-Likelihood:                -4269.0
converged:                       True   LL-Null:                       -4279.2
Covariance Type:            nonrobust   LLR p-value:                 0.0001363
```

```
==============================================================================
============
                              coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------------
Intercept                  -1.7941      0.089    -20.151      0.000      -1.969
-1.620
C(Country_HDI_rec)[T.I]     0.1124      0.138      0.814      0.416      -0.158
0.383
C(Country_HDI_rec)[T.TH]    0.3166      0.094      3.358      0.001       0.132
0.501
C(Gender)[T.une femme]      0.1148      0.058      1.984      0.047       0.001
0.228
==============================================================================
============
```

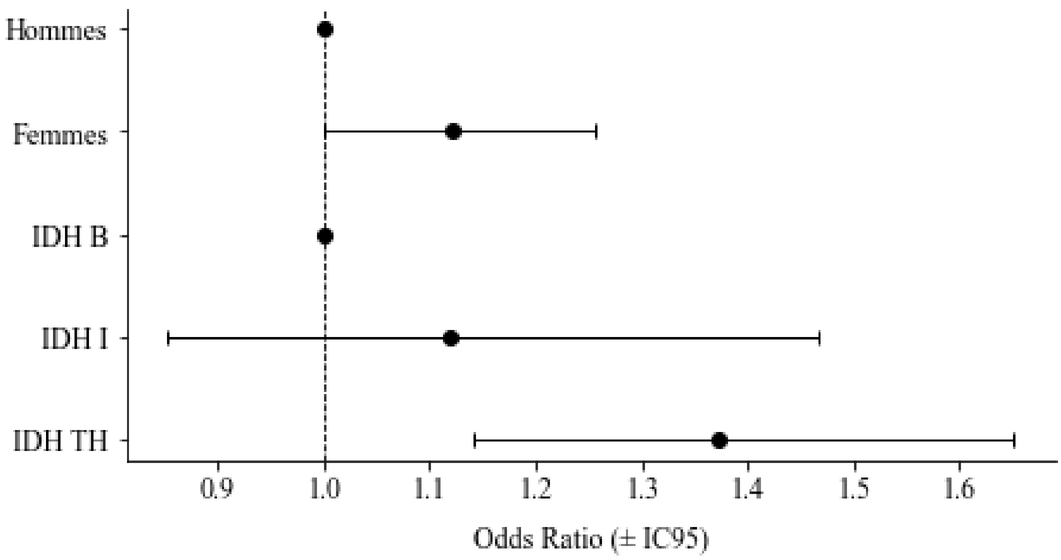
[41]:

```python
# (2) Affichage des odds-ratios
model_odds = pd.DataFrame(np.exp(my_model.params), columns= ['OR'])
model_odds['z-value']= my_model.pvalues
model_odds[['2.5%', '97.5%']] = np.exp(my_model.conf_int())
print(model_odds)
```

```
                               OR       z-value      2.5%     97.5%
Intercept                0.166282  2.617871e-90  0.139657  0.197983
C(Country_HDI_rec)[T.I]  1.118971  4.155482e-01  0.853683  1.466699
C(Country_HDI_rec)[T.TH] 1.372423  7.860490e-04  1.140860  1.650988
C(Gender)[T.une femme]   1.121606  4.721348e-02  1.001413  1.256224
```

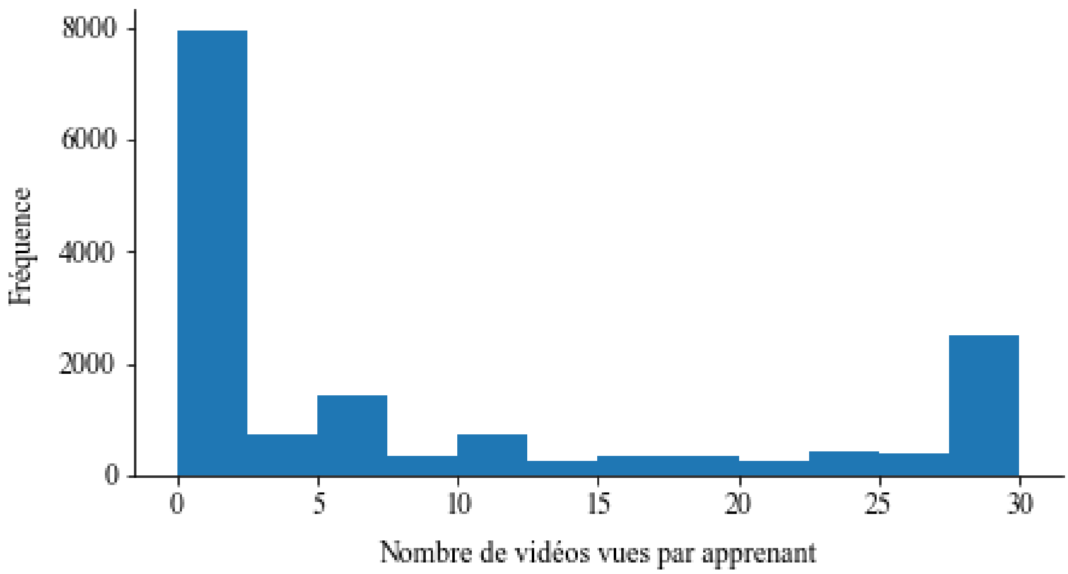
[42]:

```python
# (3) Représentation graphique des odds-ratios
plt.rc("font", family = "Times New Roman", size = 12)
plt.figure(figsize = (16.5/2.54, 9/2.54))
reference = pd.DataFrame({"OR": [1], "z-value": [0], "2.5%": [1], "97.5%":␣
 ↪[1]})
reference.index = ["Reference"]
odds_pour_graphe = pd.concat([reference, model_odds.iloc[3,:].to_frame().
 ↪transpose(), reference, model_odds.iloc[1:3,:]])
ci = [odds_pour_graphe.iloc[::-1]["OR"] - odds_pour_graphe.iloc[::-1]["2.5%"].
 ↪values, odds_pour_graphe.iloc[::-1]["97.5%"].values - odds_pour_graphe.
 ↪iloc[::-1]["OR"]]
y_labels = ["IDH TH", "IDH I", "IDH B", "Femmes", "Hommes"]
plt.errorbar(x = odds_pour_graphe.iloc[::-1]["OR"], y = y_labels, xerr = ci,␣
 ↪color = "black", capsize = 3, linestyle = "None", linewidth = 1, marker =␣
 ↪"o", markersize = 6)
plt.axvline(x = 1, linewidth = 0.8, linestyle = "--", color = "black")
plt.xlabel("Odds Ratio (± IC95)", labelpad = 8)
plt.gca().spines[["right", "top"]].set_visible(False)
plt.tight_layout()
plt.show()
```



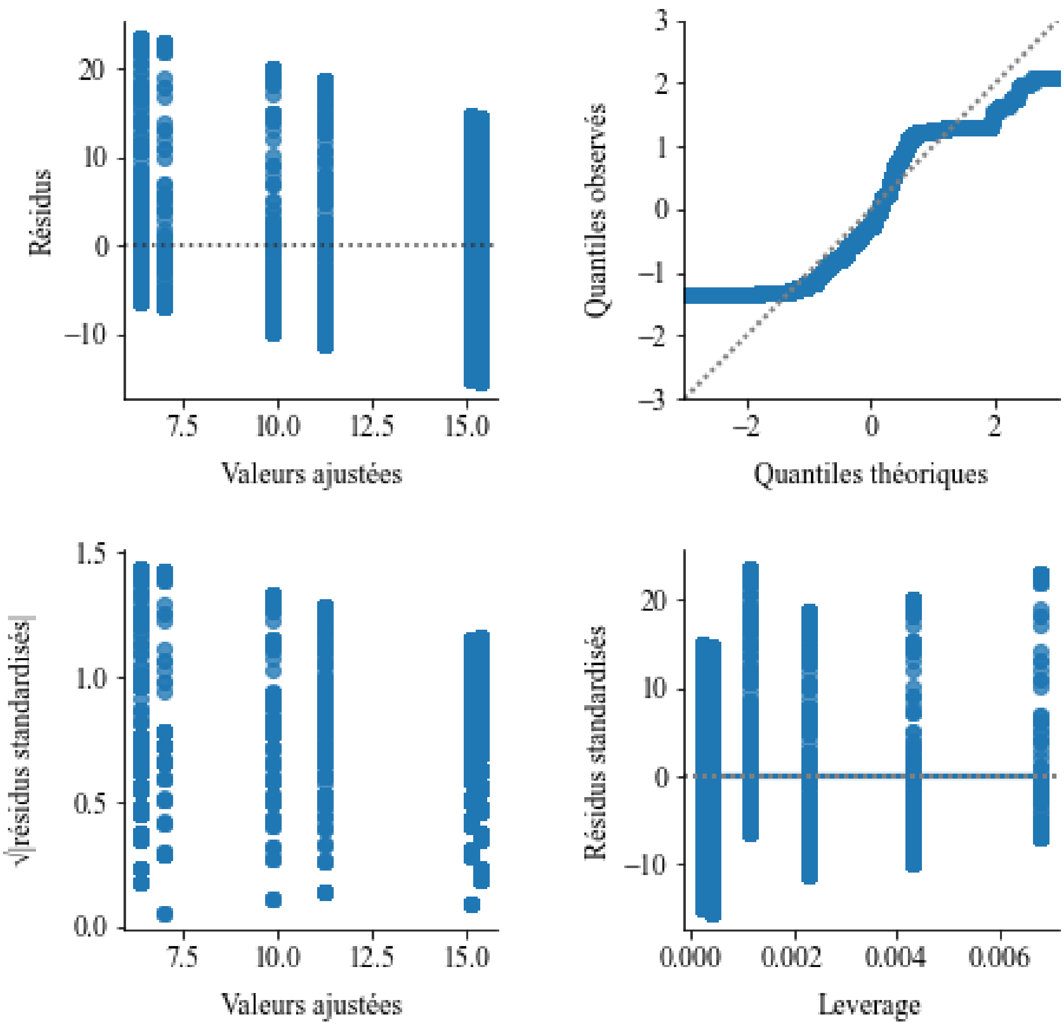
5.2 Données de comptage et loi de Poisson

[43]:
```
# (1) Distribution du nombre de vidéos vues par apprenant (sur les données␣
↪des 3 itérations)
plt.rc("font", family = "Times New Roman", size = 12)
plt.figure(figsize = (17.5/2.54, 9/2.54))
plt.hist(df_final["Nb_tot_video"], bins = 12)
plt.gca().spines[["right", "top"]].set_visible(False)
plt.ylabel("Fréquence", labelpad = 8)
plt.xlabel("Nombre de vidéos vues par apprenant", labelpad = 8)
plt.show()
```



[44]:
```
# (2) Reprise du modèle linéaire créé pour l'ANOVA (nombre de vidéos vues en␣
↪fonction du genre et de l'IDH)
my_model = ols("Nb_tot_video ~ C(Country_HDI_rec) * C(Gender)", data =␣
↪df_final).fit()
```

[45]:
```
# (3) Visualisation de l'adéquation de ce modèle
model_norm_residuals = my_model.get_influence().resid_studentized_internal
model_norm_residuals_abs_sqrt = np.sqrt(np.abs(model_norm_residuals))
summary_info = my_model.get_influence().summary_frame()
leverage = summary_info["hat_diag"]

plt.rc("font", family = "Times New Roman", size = 12)
fig, ax = plt.subplots(nrows = 2, ncols = 2, figsize = (18/2.54, 18/2.54))

sns.residplot(x = my_model.fittedvalues, y = my_model.resid, data = df_final,␣
↪ax = ax[0, 0])
ax[0, 0].set_xlabel("Valeurs ajustées", labelpad = 8)
ax[0, 0].set_ylabel("Résidus", labelpad = 2)
ax[0, 0].spines[["top", "right"]].set_visible(False)

qqplot(data = my_model.resid, fit = True, ax = ax[0, 1])
ax[0, 1].set_xlabel("Quantiles théoriques", labelpad = 8)
ax[0, 1].set_ylabel("Quantiles observés", labelpad = 10)
ax[0, 1].set_xlim(-3, 3)
ax[0, 1].set_ylim(-3, 3)
ax[0, 1].spines[["top", "right"]].set_visible(False)
ax[0, 1].axline((0, 0), slope = 1, linewidth = 2, linestyle = (0, (1, 1)),␣
↪color = "grey")

sns.regplot(x = my_model.fittedvalues, y = model_norm_residuals_abs_sqrt, ci␣
↪= None, fit_reg = False, ax = ax[1, 0])
ax[1, 0].set_xlabel("Valeurs ajustées", labelpad = 8)
ax[1, 0].set_ylabel("|résidus standardisés|", labelpad = 14)
ax[1, 0].spines[["top", "right"]].set_visible(False)

sns.regplot(x = leverage, y = my_model.resid, ci = None, fit_reg = True, ax =␣
↪ax[1, 1])
ax[1, 1].set_xlabel("Leverage", labelpad = 8)
ax[1, 1].set_ylabel("Résidus standardisés", labelpad = 4)
ax[1, 1].spines[["top", "right"]].set_visible(False)
ax[1, 1].axhline(y = 0, linewidth = 2, linestyle = (0, (1, 1)), color =␣
↪"grey")

plt.subplots_adjust(wspace = 0.5, hspace = 0.4)
```



[46]:
```
# (4) Utilisation d'un modèle plus approprié : régression de Poisson (nombre␣
↪de vidéos vues en fonction du genre et de l'IDH)
my_model = glm("Nb_tot_video ~ C(Country_HDI_rec) * C(Gender)", df_final,␣
↪family = sm.families.Poisson()).fit()
print(my_model.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:           Nb_tot_video   No. Observations:                 8951
Model:                            GLM   Df Residuals:                     8945
Model Family:                 Poisson   Df Model:                            5
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -66609.
Date:                Fri, 08 Jul 2022   Deviance:                   1.0085e+05
Time:                        19:30:07   Pearson chi2:                 8.62e+04
No. Iterations:                     5   Pseudo R-squ. (CS):             0.5189
Covariance Type:            nonrobust
================================================================================
===================================
```

```
                                                          coef    std err          z
P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------
-----------------------------------
Intercept                                               1.8522      0.013    138.951
0.000       1.826       1.878
C(Country_HDI_rec)[T.I]                                 0.5649      0.020     28.822
0.000       0.526       0.603
C(Country_HDI_rec)[T.TH]                                0.8627      0.014     62.305
0.000       0.836       0.890
C(Gender)[T.une femme]                                  0.0879      0.034      2.586
0.010       0.021       0.154
C(Country_HDI_rec)[T.I]:C(Gender)[T.une femme]         -0.2158      0.042     -5.090
0.000      -0.299      -0.133
C(Country_HDI_rec)[T.TH]:C(Gender)[T.une femme]        -0.0695      0.035     -2.011
0.044      -0.137      -0.002
====================================================================================
===================================
```