

PRÉDICTION DE REVENUS

Modèle permettant de déterminer le revenu potentiel d'une personne.

Valable pour la plupart des pays du monde.

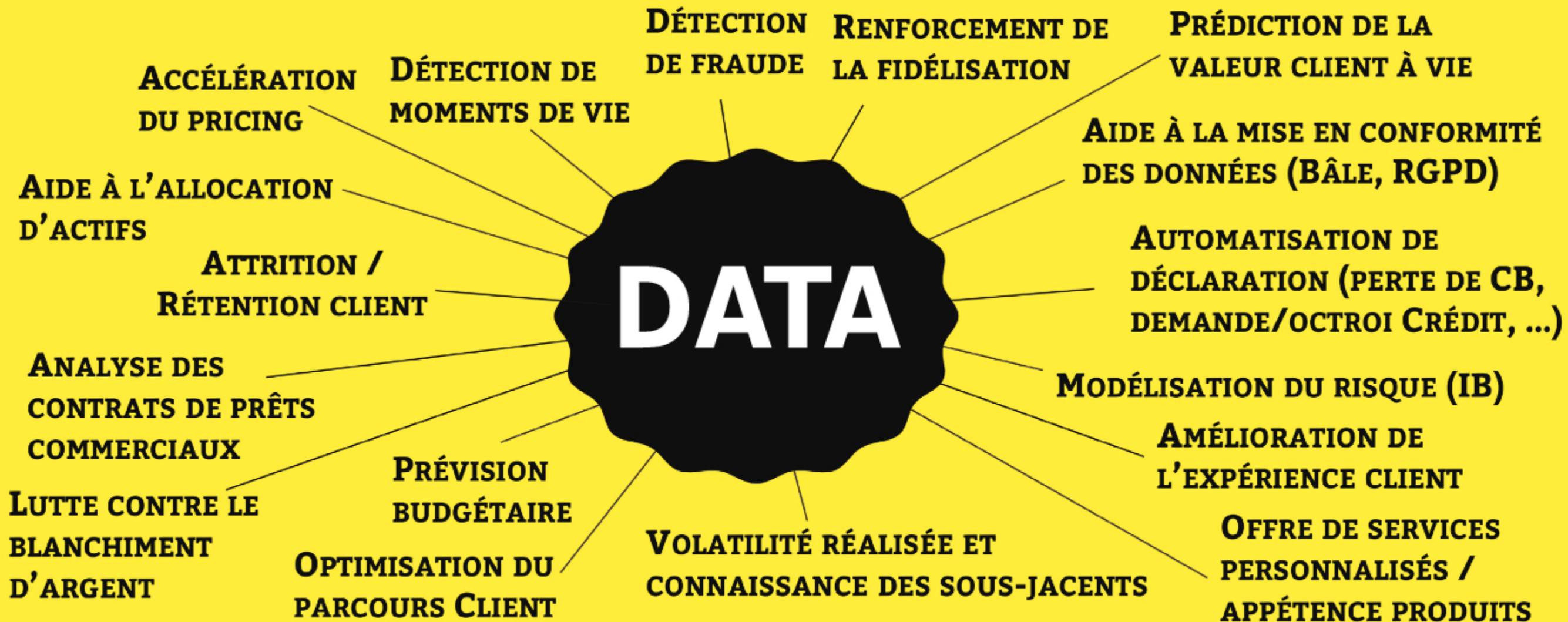
Projet 7 - Nalron Avril 2020

OpenClassrooms

ENSAE-ENSAI Formation continue

Langage utilisé Python sur Jupyter Notebook

LES BANQUES AUGMENTÉES





Sources, valeurs manquantes, outliers, doublons...

TRAITEMENT DES DONNÉES

SOURCE PRINCIPALE : WORLD INCOME DISTRIBUTION

Chargement des données : [fichier csv](#)

La population de l'échantillon a été découpée en quantiles.

- **quantile** : classes de revenu découpées en centiles
- **income** : revenu moyen des personnes d'une classe de revenu
- **gdppp** : mesure de parité du pouvoir d'achat des pays

	country	year_survey	quantile	nb_quantiles	income	gdppp
0	ALB	2008	1	100	728.89795	7297.0
1	ALB	2008	2	100	916.66235	7297.0
2	ALB	2008	3	100	1010.91600	7297.0
3	ALB	2008	4	100	1086.90780	7297.0
4	ALB	2008	5	100	1132.69970	7297.0
(11599, 6)						

La méthode des centiles permet de réduire la taille de l'échantillon, tout en préservant suffisamment d'informations pour pouvoir l'analyser.

IDENTIFICATION DE L'INDIVIDU MANQUANT

116 pays /11599 observations, **identification du quantile manquant.**

data_nunique = data.groupby('country').nunique() data_nunique[data_nunique['quantile'] < 100]						
country	year_survey	quantile	nb_quantiles	income	gdppp	country
LTU	1	1	99	1	99	1

L'observation manquante concerne le pays **Lituanie Quantile 41**

```
#Calcul du quantile manquant (41)par la moyenne des quantiles -1 et +1 (40 et 42)
ltu_41 = data.loc[(data['country'] == 'LTU') & (data['quantile'] == 40)
                  | (data['country'] == 'LTU') & (data['quantile'] == 42)]['income'].mean()
ltu_41
```

4882.14065

```
#Création de la ligne manquante selon le pays 'LTU' & quantile '41'
ltu_41_row = pd.DataFrame({'country': ['LTU'], 'year_survey': [2008], 'quantile': [41],
                           'nb_quantiles': [100], 'income': [ltu_41], 'gdppp': 17571.0})
ltu_41_row
```

	country	year_survey	quantile	nb_quantiles	income	gdppp
0	LTU	2008	41	100	4882.14065	17571.0

TRAITEMENT DES DÉSIGNATIONS PAYS

Enrichissement de l'échantillon par **les noms des pays** selon leurs codes :
source sql.sh : <https://sql.sh/514-liste-pays-csv-xml>

- Identification des codes pays manquants dans la base référentielle
- Intégration des 3 codes/pays identifiés par une méthode .concat()
- Intégration des 116 pays dans l'échantillon de travail par un merge

```
: 11 = list(df['country_code'].unique())
12 = list(country_list['country_code'].unique())
[country for country in 11 if country not in 12]

['XKX', 'MNE', 'SRB']

:#Intégration dans l'échantillon "df" des noms de pays selon leur code pays
df = pd.merge(df, country_list, how='left', on='country_code')
df = df[['country_code', 'country', 'year', 'quantile', 'nb_quantiles', 'income', 'gdppp']]
display(df.head())
display(df.shape)
```

	country_code	country	year	quantile	nb_quantiles	income	gdppp
0	ALB	Albania	2008	1	100	728.89795	7297.0
1	ALB	Albania	2008	2	100	916.66235	7297.0
2	ALB	Albania	2008	3	100	1010.91600	7297.0
3	ALB	Albania	2008	4	100	1086.90780	7297.0
4	ALB	Albania	2008	5	100	1132.69970	7297.0

TRAITEMENT DES POPULATIONS

Enrichissement de l'échantillon par les populations selon les années

source fao : <http://www.fao.org/faostat/fr/#data/OA>

- Identification des pays avec différence de désignation
- Traitement des écarts par le développement d'une fonction spécifique
- Jointure des données pour intégration dans l'échantillon

```
#Fonction de retraitement des noms de pays exposant des écarts d'intitulé
def rename_country(country):
    if (country == 'Bolivia (Plurinational State of)'):
        return 'Bolivia'
    elif country == 'Central African Republic':
        return 'Central African'
    elif country == 'Czechia':
        return 'Czech Republic'
    elif country == 'Iran (Islamic Republic of)':
        return 'Islamic Republic of Iran'
    elif country == 'North Macedonia':
        return 'The Former Yugoslav Republic of Macedonia'
    elif country == 'Sudan (former)':
        return 'Sudan'
    elif country == 'Eswatini':
        return 'Swaziland'
    elif country == 'China, Taiwan Province of':
        return 'Taiwan'
    elif country == 'United Republic of Tanzania':
        return 'United Republic Of Tanzania'
    elif country == 'United States of America':
        return 'United States'
    elif country == 'Venezuela (Bolivarian Republic of)':
        return 'Venezuela'
    elif country == 'Viet Nam':
        return 'Vietnam'
    elif country == 'Palestine':
        return 'Occupied Palestinian Territory'
    elif country == 'Congo':
        return 'The Democratic Republic Of The Congo'
    else :
        return country
```

	country_code	country	year	quantile	nb_quantiles	income	gdppp	population
0	ALB	Albania	2008	1	100	728.89795	7297.0	3002678.0
1	ALB	Albania	2008	2	100	916.66235	7297.0	3002678.0
2	ALB	Albania	2008	3	100	1010.91600	7297.0	3002678.0
3	ALB	Albania	2008	4	100	1086.90780	7297.0	3002678.0
4	ALB	Albania	2008	5	100	1132.69970	7297.0	3002678.0

(11600, 8)

	year	country_quantity	population	pop_world	%world_pop
0	2004		1	17827825	6432374971
1	2006		5	287548000	6593623202
2	2007		15	2750266740	6675130418
3	2008		76	2271976191	6757887172
4	2009		12	475559459	6840591577
5	2010		6	383832444	6922947261
6	2011		1	14948801	7004011262

INTÉGRATION DES INDICES DE GINI

Enrichissement de l'échantillon par **les indices de Gini des pays** :
source world bank : <https://data.worldbank.org/indicator/SI.POV.GINI>

- Identification des indices de Gini selon le couple Pays/Années
- Jointure et traitement / calcul des indices manquants

```
#Intégration des valeurs manquantes dans l'échantillon de travail "df"
for code in list(find_gini['country_code'].unique()):
    df.loc[df['country_code'] == code, 'gini'] = (find_gini.loc[find_gini['country_code'] == code, 'gini'])

#Visualisation rapide des données
df
```

	country_code	country	year	quantile	nb_quantiles	income	gdppp	population	gini
0	ALB	Albania	2008	1	100	728.89795	7297.00000	3002678.0	0.30
1	ALB	Albania	2008	2	100	916.66235	7297.00000	3002678.0	0.30
2	ALB	Albania	2008	3	100	1010.91600	7297.00000	3002678.0	0.30
3	ALB	Albania	2008	4	100	1086.90780	7297.00000	3002678.0	0.30
4	ALB	Albania	2008	5	100	1132.69970	7297.00000	3002678.0	0.30
...
11595	COD	The Democratic Republic Of The Congo	2008	97	100	911.78340	303.19305	4011486.0	0.44
11596	COD	The Democratic Republic Of The Congo	2008	98	100	1057.80740	303.19305	4011486.0	0.44
11597	COD	The Democratic Republic Of The Congo	2008	99	100	1286.60290	303.19305	4011486.0	0.44
11598	COD	The Democratic Republic Of The Congo	2008	100	100	2243.12260	303.19305	4011486.0	0.44
11599	LTU	Lithuania	2008	41	100	4882.14065	17571.0

11600 rows x 9 columns

```
#Calcul des indices de gini manquants
list_gini = []
for code in missing_gini['country_code']:
    dep = missing_gini[missing_gini['country_code'] == code]['income'].values
    n = len(dep)
    lorenz = np.cumsum(np.sort(dep)) / dep.sum()

    AUC = (lorenz.sum() - lorenz[-1]/2 - lorenz[0]/2)/n
    S = 0.5 - AUC
    gini = round(2*S, 2)
    list_gini.append(gini)
```

BRIEF SUR LA DISTRIBUTION DES VARIABLES

Visualisation de la **distribution des variables**.

- Identification de quelques outliers :

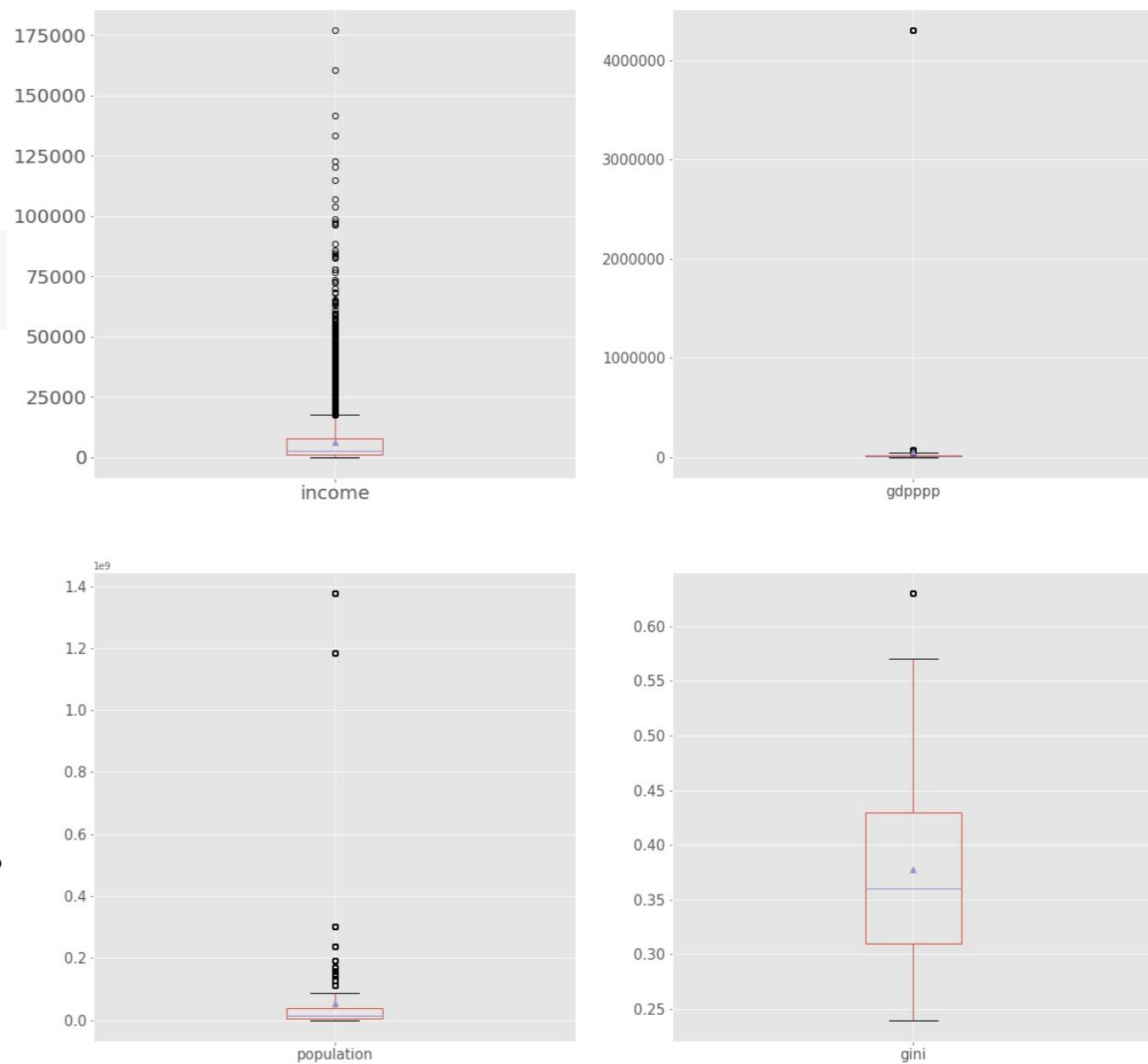
```
display(df.sort_values(by='population', ascending=False).iloc[:1, :])
display(df.sort_values(by='gdppp', ascending=False).iloc[:1, :])
display(df.sort_values(by='gini', ascending=False).iloc[:1, :])
```

	country_code	country	year	quantile	nb_quantiles	income	gdppp	population	gini
1798	CHN	China	2007	99	100	11071.51	5712.0	1.376266e+09	0.48

	country_code	country	year	quantile	nb_quantiles	income	gdppp	population	gini
3200	FJI	Fiji	2008	1	100	308.17334	4300332.0	845361.0	0.4

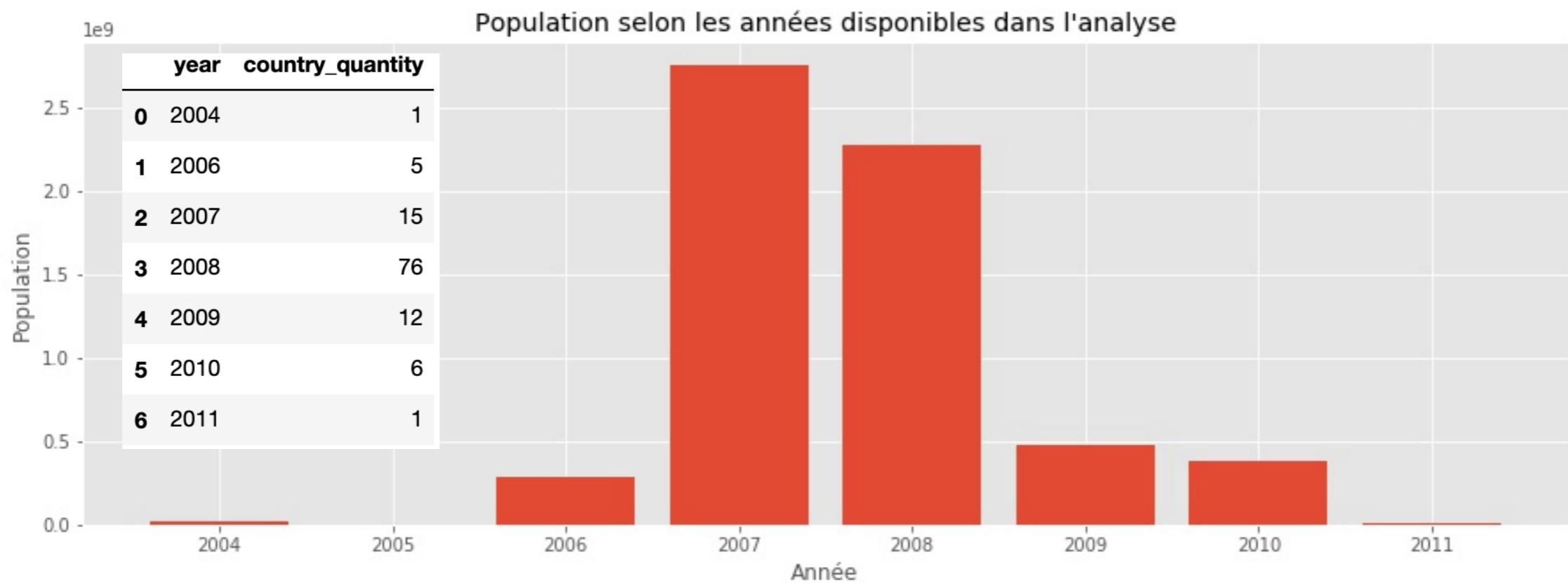
	country_code	country	year	quantile	nb_quantiles	income	gdppp	population	gini
11400	ZAF	South Africa	2008	2	100	138.34155	9602.0	49779471.0	0.63

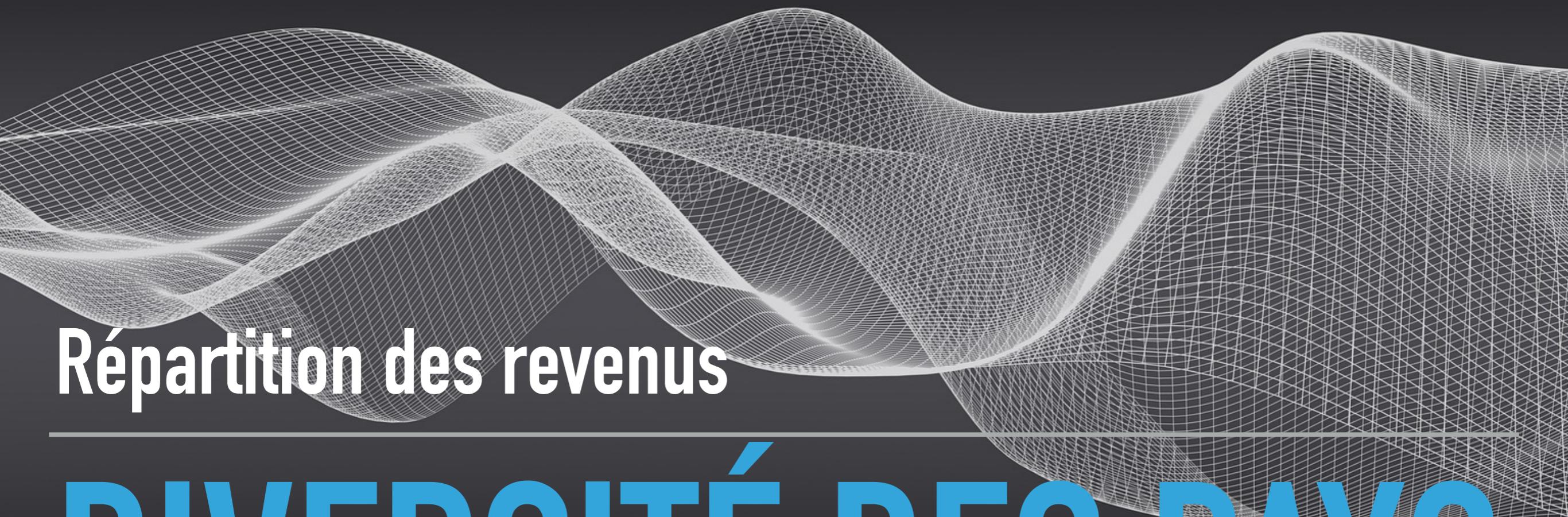
- Aucune valeur doublon identifiée.



QUELQUES CHIFFRES...

Années utilisées dans l'analyse de 2004 à 2011 (2005 non représentée).
Forte concentration sur les années **2007 et 2008**.
116 pays représentés dans l'échantillon.





Répartition des revenus

DIVERSITÉ DES PAYS

CALCUL DU REVENU MOYEN DU PAYS DANS LEQUEL HABITE LE PROSPECT

Le **revenu moyen du pays** est nécessaire pour aller plus loin dans la problématique.
Intégration d'une variable supplémentaire **income_avg** pour chaque pays :

```
: df_income_avg = df[['country_code','income']].groupby('country_code').mean().reset_index()
df_income_avg.rename(columns={'income':'income_avg'}, inplace=True)
df = df.merge(df_income_avg, how='left', on='country_code')
display(df.shape)
display(df.head())
```

(11600, 10)

	country_code	country	year	quantile	nb_quantiles	income	gdppp	population	gini	income_avg
0	ALB	Albania	2008	1	100	728.89795	7297.0	3002678.0	0.3	2994.829902
1	ALB	Albania	2008	2	100	916.66235	7297.0	3002678.0	0.3	2994.829902
2	ALB	Albania	2008	3	100	1010.91600	7297.0	3002678.0	0.3	2994.829902
3	ALB	Albania	2008	4	100	1086.90780	7297.0	3002678.0	0.3	2994.829902
4	ALB	Albania	2008	5	100	1132.69970	7297.0	3002678.0	0.3	2994.829902

Le but est de montrer la diversité des pays en termes de distribution de revenus à l'aide d'un graphique.

Comment représenter graphiquement le revenu moyen de chacune des classes de revenus pour 5 à 10 pays choisis pour montrer la diversité des cas?

CLUSTERING KMEANS

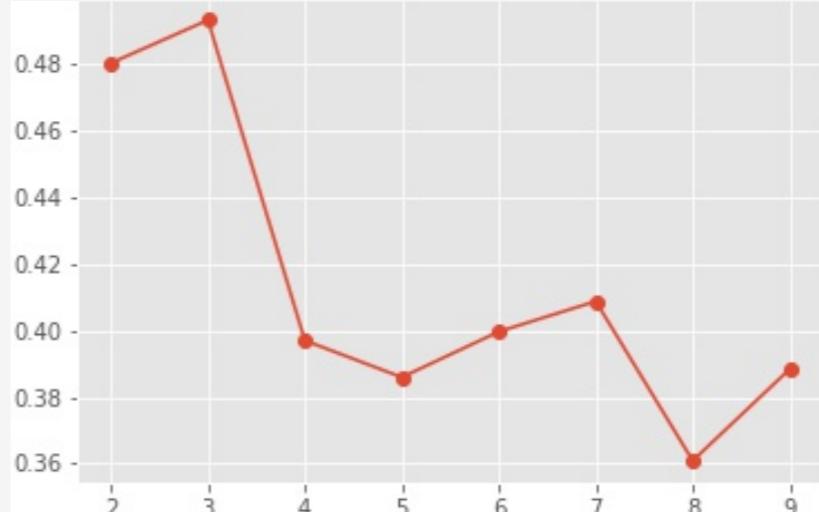
Par simplification le **clustering** sera fait sur **le premier quantile** sur les variables '**gdpppp**' et '**gini**'

```
#Calcul de la métrique "silhouette" pour différents nombres de groupes issus de la méthode des centres mobiles
#Liste pour stocker nos coefficients
silhouettes = []

#Boucle itérative de 2 à 10 (clusters) pour tester les possibilités
for k in range(2, 10):
    #Création et ajustement d'un modèle pour chaque k
    cls = cluster.KMeans(n_clusters=k)
    cls.fit(X_scaled)

    #Stockage des coefficients associés
    silh = metrics.silhouette_score(X_scaled, cls.labels_)
    silhouettes.append(silh)

#Visualisation des valeurs de coefficient de silhouette pour chaque nombre de cluster
plt.plot(range(2, 10), silhouettes, marker='o')
plt.savefig('p7_03_graphic/country_kmeans.jpg')
plt.show()
```



k	Silhouette Score
2	0.48
3	0.485
4	0.40
5	0.395
6	0.405
7	0.41
8	0.36
9	0.39

k=3 semble pertinent

ce coefficient de silhouette est le plus élevé, le plus représentatif.

CLUSTERING KMEANS : CHOIX DU CLUSTER

Choix des pays du **cluster 2**, car le gdpppp est plus élevé avec un indice de Gini très éloigné de 1 indiquant une meilleure répartition des revenus.

#Liste des pays retenus

```
country_list = ['Luxembourg', 'Sweden', 'Spain', 'France', 'Croatia', 'Norway', 'United States', 'Denmark', 'Austria']
country_code = ['LUX', 'SWE', 'ESP', 'FRA', 'HRV', 'NOR', 'USA', 'DNK', 'AUT']
```

#Centroïdes des 3 clusters

	gdpppp	gini
cluster0	-0.464604	1.222659
cluster1	-0.468108	-0.486949
cluster2	1.597383	-0.766041

#Pays proches des centroides

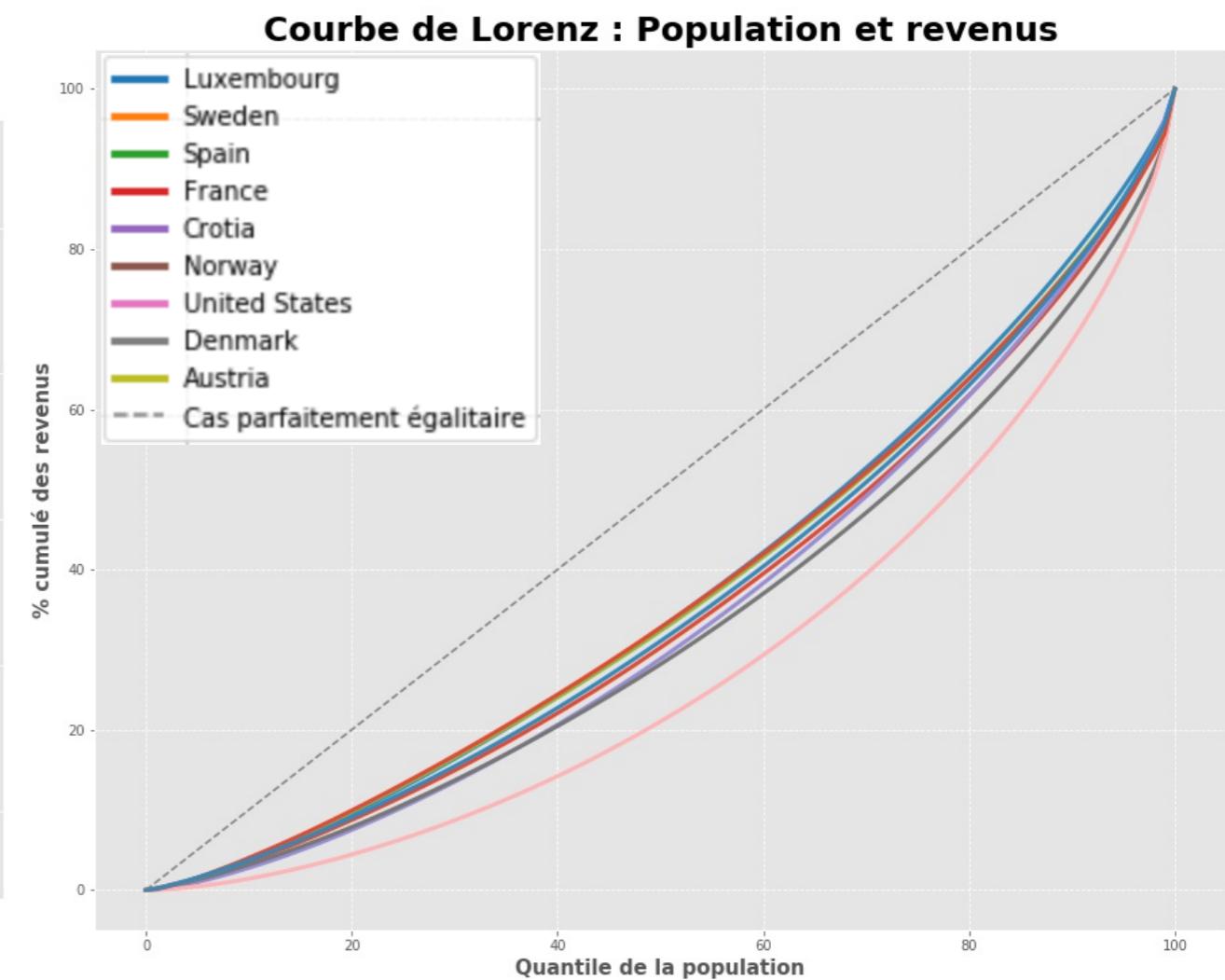
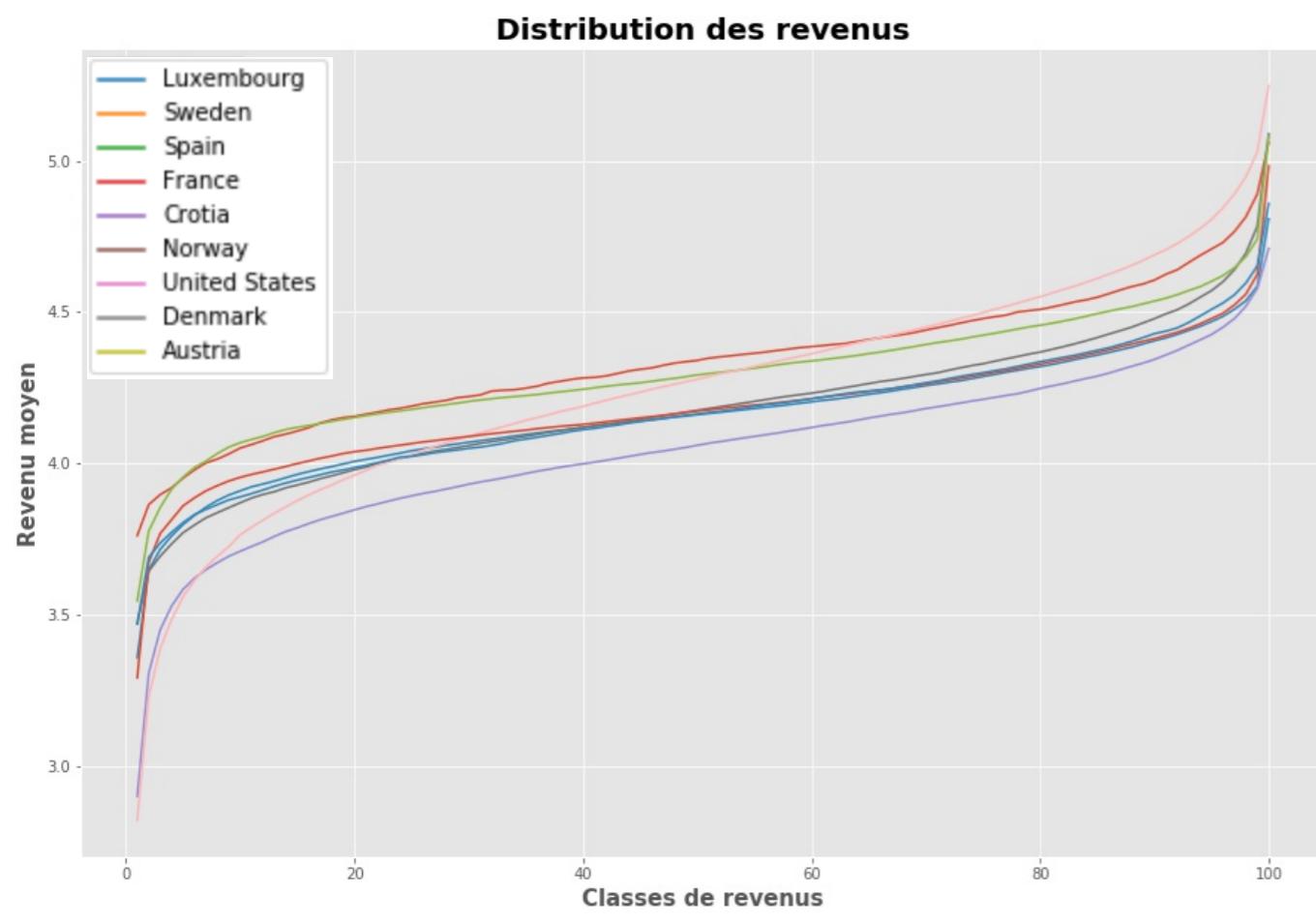
country	country_code	year	quantile	nb_quantiles	income	gdpppp	population	gini	income_avg
Luxembourg	LUX	2008	1	100	5780.837400	73127.0	485405.0	0.33	25217.562681
Sweden	SWE	2008	1	100	2284.432900	34371.0	9236428.0	0.28	16184.222707
Spain	ESP	2008	1	100	796.719060	28336.0	46068811.0	0.34	13116.992910
Estonia	EST	2008	1	100	959.560100	18773.0	1340073.0	0.32	7702.062593
Venezuela	VEN	2006	1	100	171.892820	11756.0	26850194.0	0.43	3167.147724
Costa Rica	CRI	2008	1	100	275.653900	10374.0	4463125.0	0.49	5580.386556
Kosovo	XKX	2008	1	100	437.893700	7249.0	1747383.0	0.31	2176.269035
Guatemala	GTM	2011	1	100	38.463615	4367.0	14948801.0	0.57	2142.474753
Nicaragua	NIC	2009	1	100	53.809303	2576.0	5745526.0	0.44	2519.333262
Yemen	YEM	2008	1	100	162.951310	2224.0	21892146.0	0.37	1042.635870

#Pays cluster 2 fortement contributeurs

c_inertie	country	country_code	cluster	gdpppp	gini
62	21.617245	Luxembourg	LUX	1	73127.0 0.33
80	9.452728	Norway	NOR	1	49070.0 0.27
108	5.900455	United States	USA	1	43261.0 0.43
25	5.113030	Denmark	DNK	1	34130.0 0.25
79	4.915114	Netherlands	NLD	1	38065.0 0.29
44	4.821413	Ireland	IRL	1	39268.0 0.31
96	4.173884	Sweden	SWE	1	34371.0 0.28
3	4.136935	Austria	AUT	1	36193.0 0.30
95	4.046596	Slovenia	SVN	1	27197.0 0.24
31	3.988318	Finland	FIN	1	33626.0 0.28
5	3.972432	Belgium	BEL	1	33561.0 0.28
47	3.830395	Iceland	ISL	1	36527.0 0.32
24	3.288571	Germany	DEU	1	33758.0 0.31
15	3.281590	Canada	CAN	1	35895.0 0.35
33	2.896048	United Kingdom	GBR	1	34048.0 0.34
94	2.774019	Slovakia	SVK	1	20515.0 0.25
23	2.705239	Czech Republic	CZE	1	23223.0 0.26
51	2.535107	Japan	JPN	1	31307.0 0.32
32	2.178491	France	FRA	1	30357.0 0.33
29	1.658870	Spain	ESP	1	28336.0 0.34

ANALYSE GRAPHIQUE DE LA DISTRIBUTION DES REVENUS

Suède et Danemark présentent une distribution des revenus au sein de la population beaucoup plus égalitaire que les autres pays.
Etats-Unis et la France sont à l'opposé, on peut distinguer une distribution de revenus plus inégalitaire.



QUELLE ÉVOLUTION POUR LES INDICES DE GINI ?

On peut noter une baisse pour la France, Espagne et Suède :
un peu plus d'inégalité entre les catégories sociales.

Position de la France

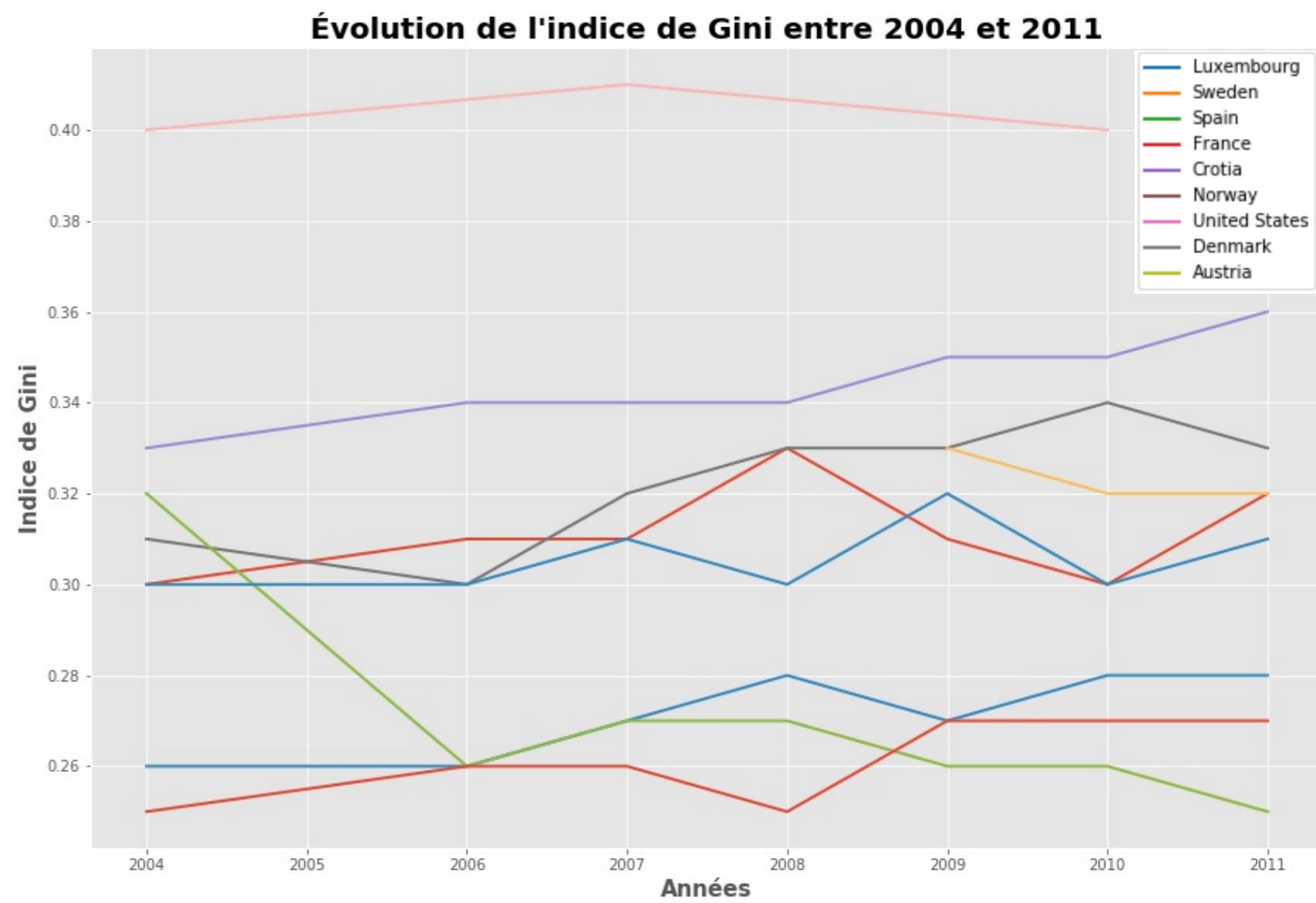
country_code	country	gini	rang
45	FRA	France	0.322857

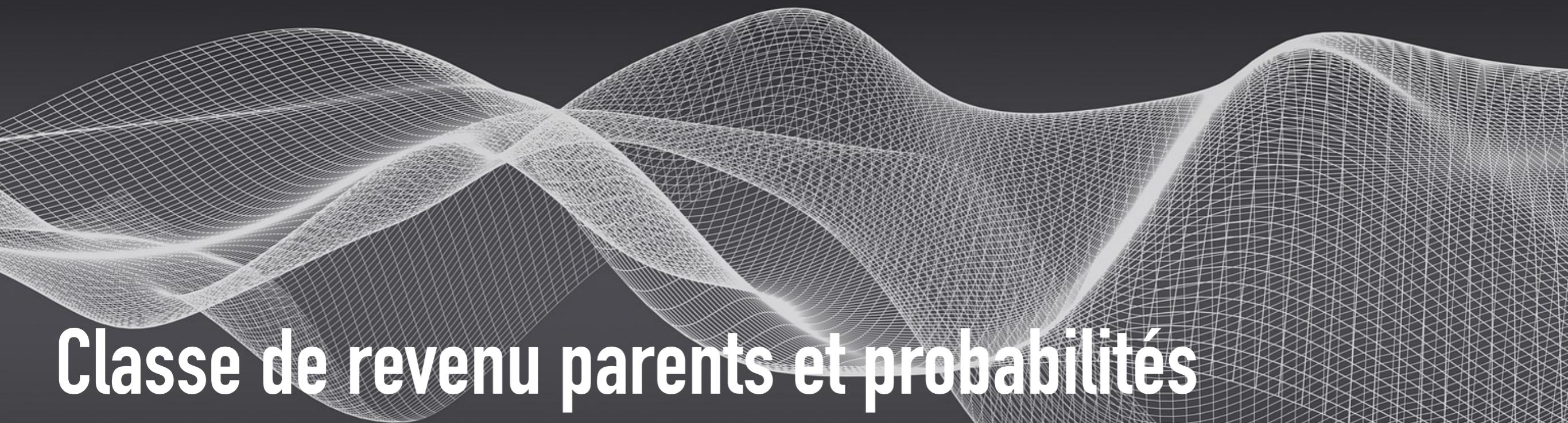
5 pays : indice Gini + faible

country_code	country	gini	rang
120	SVN	Slovenia	0.245714
35	DNK	Denmark	0.261429
119	SVK	Slovak Republic	0.262857
33	CZE	Czech Republic	0.265714
135	UKR	Ukraine	0.268571

5 pays : indice Gini + fort

country_code	country	gini	rang
143	ZAF	South Africa	0.63
94	NAM	Namibia	0.61
18	BWA	Botswana	0.60
29	COM	Comoros	0.56
19	CAF	Central African Republic	0.56





Classe de revenu parents et probabilités

GÉNÉRATION DES REVENUS

COMMENT CALCULER LA CLASSE DE REVENU DES PARENTS ?

A ce stade de l'étude nous avons deux des trois variables explicatives : **revenu moyen du pays, l'indice de Gini du pays**

La classe de revenu parent peut-être simulée avec le coef d'élasticité [+info sources multiples](#) : - extrapolation par région du monde [élasticité.txt](#)
- banque mondiale dans [GDIM dataset](#)

Boucle itérative afin d'imputer les valeurs NaN sur la variable 'IGEincome'

```
for code in missing_countries_gdim :
    if list(gdim.loc[gdim['country_code'] == code, 'region']) == ['South Asia']:
        gdim.loc[gdim['country_code'] == code, 'IGEincome'] = 0.50
    elif list(gdim.loc[gdim['country_code'] == code, 'region']) == ['Sub-Saharan Africa']:
        gdim.loc[gdim['country_code'] == code, 'IGEincome'] = 0.66
    elif list(gdim.loc[gdim['country_code'] == code, 'region']) == ['Latin America & Caribbean']:
        gdim.loc[gdim['country_code'] == code, 'IGEincome'] = 0.66
    elif list(gdim.loc[gdim['country_code'] == code, 'region']) == ['Europe & Central Asia']:
        gdim.loc[gdim['country_code'] == code, 'IGEincome'] = 0.40
    elif list(gdim.loc[gdim['country_code'] == code, 'region']) == ['East Asia & Pacific']:
        gdim.loc[gdim['country_code'] == code, 'IGEincome'] = 0.50
    else :
        gdim.loc[gdim['country_code'] == code, 'IGEincome'] = 0.40
```

	country	country_code	region	IGEincome
0	Afghanistan	AFG	South Asia	NaN
12	Angola	AGO	Sub-Saharan Africa	NaN
60	Albania	ALB	Europe & Central Asia	0.815874
84	Argentina	ARG	Latin America & Caribbean	NaN
99	Armenia	ARM	Europe & Central Asia	NaN
183	Australia	AUS	High income	0.275000
243	Austria	AUT	High income	0.245267
279	Azerbaijan	AZE	Europe & Central Asia	NaN
351	Belgium	BEL	High income	0.183176
405	Benin	BEN	Sub-Saharan Africa	0.855116
(150, 4)				

Puis matching par jointure avec l'échantillon.

GÉNÉRATION ALÉATOIRE DES CLASSES DE REVENU DES PARENTS

A partir du **coeffcient d'élasticité** et la **classe de revenu de l'enfant**, nous pouvons générer aléatoirement une **classe de revenu des parents**.

Le protocole de génération des données est le suivant :

- > échantillon supérieur à 1000 fois le nombre de quantiles.
- > calcul du revenu enfant : $y_{child} = e^{\alpha + p_j \ln(y_{parent}) + \epsilon}$
- > pour un coef. d'élasticité donné, ici celui de la France (0.33)
- > calcul des classes de revenu enfants et parents
- > estimation des distributions conditionnelles d'avoir ces classes parents

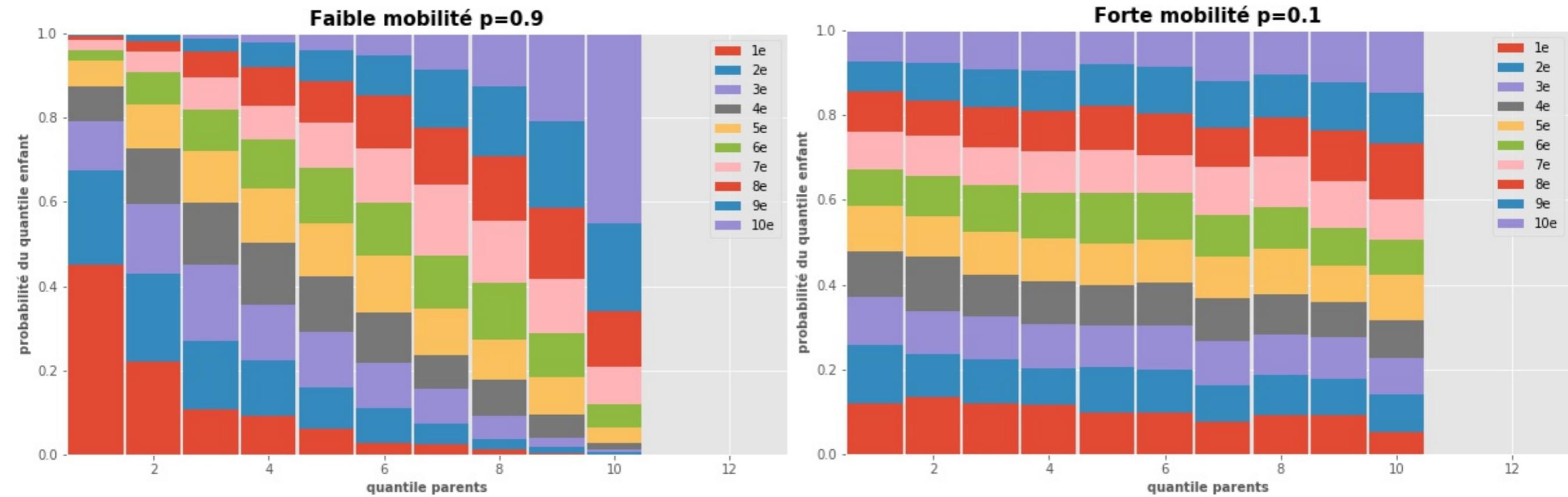
	y_child	y_parents	c_i_child	c_i_parent
0	0.589664	0.664054	31	35
1	1.268625	3.213953	59	88
2	5.368220	0.442272	95	21
3	0.706779	1.482513	37	66
4	1.024113	0.642365	51	33
(100000, 4)				

	y_child	y_parents	c_i_child	c_i_parent	proba
0	0.589664	0.664054	31	35	0.007
1	1.268625	3.213953	59	88	0.005
2	5.368220	0.442272	95	21	0.003
3	0.706779	1.482513	37	66	0.008
4	1.024113	0.642365	51	33	0.007

VISUALISATION DES DISTRIBUTIONS CONDITIONNELLES

A partir de la matrice générée, on visualise **deux cas extrêmes de mobilité**
exemple : 10ème classe parents avec moins de 50% de mobilité.

> ces enfants sont majoritairement dans la classe de revenu des parents.



GÉNÉRATION D'UN NOUVEL ÉCHANTILLON 500X PLUS GRAND

Création d'un clone, à partir de la WID, et selon les règles conditionnelles.

> **Attribution des classes parents aux 500 individus de chaque pays.**

#Matrice des distributions conditionnelles

```
%%time
nb_quantiles = 100
n = 100*500
start_time = time.time()

list_cd = {}
for i, row in enumerate(country_income.iterrows()):
    print(f"""{time.time()-start_time:.2f}s - pays {i+1}/{len(country_income)}""")
    y_child, y_parents = generate_incomes(n, row[1][0])
    sample_final = compute_quantiles(y_child, y_parents, nb_quantiles)
    cd = conditional_distributions(sample_final, nb_quantiles)
    list_cd[row[0]] = cd.tolist()
```

#Boucle itérative sur chaque pays

```
%%time
#pour chaque pays du dataframe final
df_wid500['c_k_parent'] = NaN

for i, code_pays in enumerate(df_income_mobility['country_code'].drop_duplicates().values):
    #seconde boucle sur chaque classe de revenu
    for c_parents in np.arange(1,101, dtype='int8'):
        #Récupération des distributions de la liste 'list_cd'
        dc = list_cd[code_pays][c_parents-1]

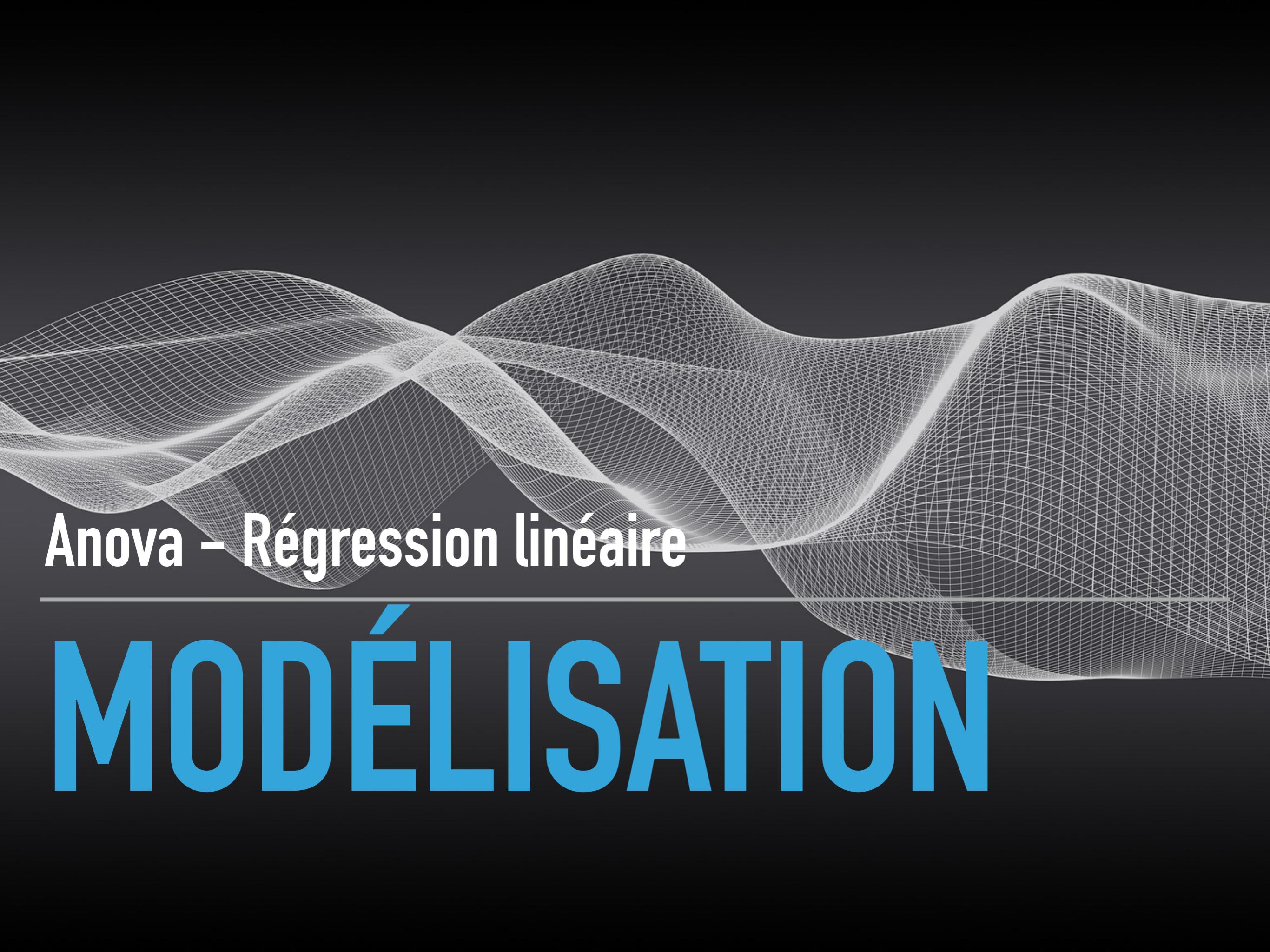
        #calcul des valeurs en fonction de cette distribution
        c_k_parent = []
        for i in range(len(dc)):
            classe_parent = int(dc[i]*500)
            c_k_parent.extend([i+1]*classe_parent)

        #affectation des valeurs aux individus de l'échantillon
        df_wid500.loc[(df_wid500['country_code'] == code_pays) &
                      (df_wid500['quantile'] == c_parents), 'c_k_parent'] = c_k_parent
```

#Aperçu de l'échantillon final

(5800000, 8)

	country_code	country	quantile	c_k_parent	population	income	income_avg	gini
0	ALB	Albania	1	1	3002678.0	728.89795	2994.829902	0.3
1	ALB	Albania	2	1	3002678.0	916.66235	2994.829902	0.3
2	ALB	Albania	3	1	3002678.0	1010.91600	2994.829902	0.3
3	ALB	Albania	4	1	3002678.0	1086.90780	2994.829902	0.3
4	ALB	Albania	5	1	3002678.0	1132.69970	2994.829902	0.3



Anova – Régression linéaire

MODÉLISATION

ANOVA : PAYS ~ REVENU MOYEN DES INDIVIDUS

L'ANOVA sur le pays explique **50% de la variance** du revenu de l'individu.

```
anova_income = smf.ols('income ~ country_code', data=df_wid500_final).fit()
print(anova_income.summary().tables[0])
```

```
OLS Regression Results
=====
Dep. Variable:           income    R-squared:                 0.496
Model:                  OLS        Adj. R-squared:            0.496
Method:                 Least Squares   F-statistic:             4.971e+04
Date:      Tue, 24 Mar 2020   Prob (F-statistic):       0.00
Time:      12:04:14          Log-Likelihood:          -5.9310e+07
No. Observations:      5800000    AIC:                      1.186e+08
Df Residuals:          5799884   BIC:                      1.186e+08
Df Model:                   115
Covariance Type:        nonrobust
=====
```

L'ANOVA du logarithme du revenu explique **73% de la variance**.

```
#Anova logarithmique
anova_ln_income = smf.ols('ln_income ~ country_code', data=df_wid500_final).fit()
print(anova_ln_income.summary().tables[0])
```

```
OLS Regression Results
=====
Dep. Variable:           ln_income    R-squared:                 0.729
Model:                  OLS        Adj. R-squared:            0.729
Method:                 Least Squares   F-statistic:             1.358e+05
Date:      Tue, 24 Mar 2020   Prob (F-statistic):       0.00
Time:      12:06:55          Log-Likelihood:          -6.3135e+06
No. Observations:      5800000    AIC:                      1.263e+07
Df Residuals:          5799884   BIC:                      1.263e+07
Df Model:                   115
Covariance Type:        nonrobust
=====
```

RÉGRESSION LINÉAIRE 1ER MODÈLE : REVENU MOYEN PAYS ~ INDICE GINI

Ce modèle n'explique que **50% de la variance**, équivalent à l'ANOVA.

Problème de linéarité?

Les salaires ont tendance à évoluer de manière exponentielle.

L'analyse de ce modèle sera faite plus bas pour vérifier cette hypothèse.

```
#Création du premier modèle de Régression linéaire
modele1 = smf.ols('income ~ gini + income_avg', data=df_wid500_final).fit()
print(modele1.summary())
```

OLS Regression Results						
Dep. Variable:	income	R-squared:	0.496			
Model:	OLS	Adj. R-squared:	0.496			
Method:	Least Squares	F-statistic:	2.858e+06			
Date:	Tue, 24 Mar 2020	Prob (F-statistic):	0.00			
Time:	12:11:01	Log-Likelihood:	-5.9310e+07			
No. Observations:	5800000	AIC:	1.186e+08			
Df Residuals:	5799997	BIC:	1.186e+08			
Df Model:	2					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-3.829e-09	14.870	-2.57e-10	1.000	-29.144	29.144
gini	4.451e-09	35.858	1.24e-10	1.000	-70.280	70.280
income_avg	1.0000	0.000	2264.694	0.000	0.999	1.001
Omnibus:	7299083.278	Durbin-Watson:	0.685			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2103715771.080			
Skew:	6.739	Prob(JB):	0.00			
Kurtosis:	95.322	Cond. No.	1.25e+05			

RÉGRESSION LINÉAIRE 2EME MODÈLE : EN LOGARITHME

Ce modèle logarithmique explique **73%** de la variance !

Les p-valeurs sont d'ailleurs très faibles, au final on obtient les mêmes performances qu'avec l'**ANOVA du logarithme du revenu.**

```
#Création d'un second modèle de Régression linéaire à des fins de comparaison selon le logarithme
modele2 = smf.ols('ln_income ~ gini + ln_income_avg', data=df_wid500_final).fit()
print(modele2.summary())
```

```

=====
                         OLS Regression Results
=====
Dep. Variable:          ln_income    R-squared:                 0.728
Model:                  OLS         Adj. R-squared:            0.728
Method:                Least Squares   F-statistic:             7.774e+06
Date:      Tue, 24 Mar 2020   Prob (F-statistic):        0.00
Time:      12:13:38           Log-Likelihood:          -6.3234e+06
No. Observations:      5800000    AIC:                      1.265e+07
Df Residuals:          5799997    BIC:                      1.265e+07
Df Model:                   2
Covariance Type:       nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.4670	0.003	161.378	0.000	0.461	0.473
gini	-1.7367	0.004	-460.184	0.000	-1.744	-1.729
ln_income_avg	0.9903	0.000	3685.515	0.000	0.990	0.991

	377586.682	Durbin-Watson:	0.388
Omnibus:	0.000	Jarque-Bera (JB):	1774036.803
Prob(Omnibus):		Prob(JB):	0.00
Skew:	-0.097	Cond. No.	121.
Kurtosis:	5.702		

AMÉLIORATION DU MODÈLE AVEC LA CLASSE DE REVENU PARENTS

Ce modèle **monte en performance de 5 points** par rapport au précédent.

Le coefficient de détermination atteint 0.778 (contre 0.728).

22% de la variance n'est pas expliquée : autres caractéristiques possibles comme l'âge, le sexe, le niveau de qualification, etc...

```
#Modèle de Régression linéaire avec ajout d'une variable supplémentaire
modele2_parents = smf.ols('ln_income ~ gini + c_k_enfants + ln_income_avg', data=df_wid500_final).fit()
print(modele2_parents.summary())
```

```

=====
                         OLS Regression Results
=====
Dep. Variable:          ln_income    R-squared:                 0.778
Model:                  OLS         Adj. R-squared:            0.778
Method:                Least Squares   F-statistic:             6.791e+06
Date:                  Fri, 10 Apr 2020   Prob (F-statistic):      0.00
Time:                  10:19:09        Log-Likelihood:           -5.7325e+06
No. Observations:      5800000        AIC:                      1.147e+07
Df Residuals:          5799996        BIC:                      1.147e+07
Df Model:                   3
Covariance Type:       nonrobust
=====
```

DÉCOMPOSITION DE LA VARIANCE TOTALE EXPLIQUÉE

Formule de décomposition de la variance **SCT = SCE + SCR**

#SCT

```
regr_log_sct = res_regr_log.values[:,0].sum()      10153966.687247703  
regr_log_sct
```

#SCE

```
regr_log_sce = res_regr_log.values[0,0]+res_regr_log.values[1,0]+res_regr_log.values[2,0]  7702752.0785522945  
regr_log_sce
```

#SCR

```
regr_log_scr = res_regr_log.values[2,0]    7038685.026525856  
regr_log_scr
```

#Variance expliquée par le revenu moyen du pays (ln_income_avg)

```
regr_log_sce_incomeavg = sm.stats.anova_lm(modele2_parents, typ=2).values[2,0]/regr_log_sct  0.6931955996434076  
regr_log_sce_incomeavg
```

#Variance expliquée par l'indice de Gini

```
regr_log_sce_gini = sm.stats.anova_lm(modele2_parents, typ=2).values[0,0]/regr_log_sct  0.010807426696892164  
regr_log_sce_gini
```

#Variance expliquée par la classe de revenu des parents

```
regr_log_sce_pa = sm.stats.anova_lm(modele2_parents, typ=2).values[1,0]/regr_log_sct  0.054592339963799724  
regr_log_sce_pa
```

Le revenu moyen du pays joue un rôle majeur avec plus de 69%, tandis que la classe de revenu des parents n'explique 5.5%, puis l'indice de Gini à 1%.



Leviers, résidus, distance de cook...

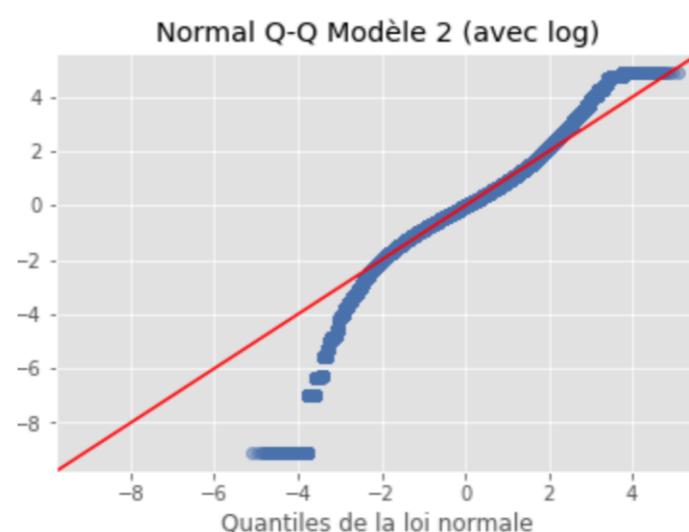
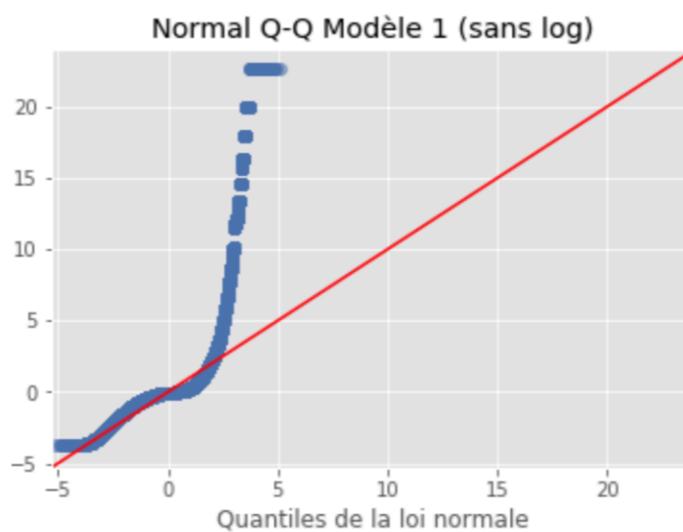
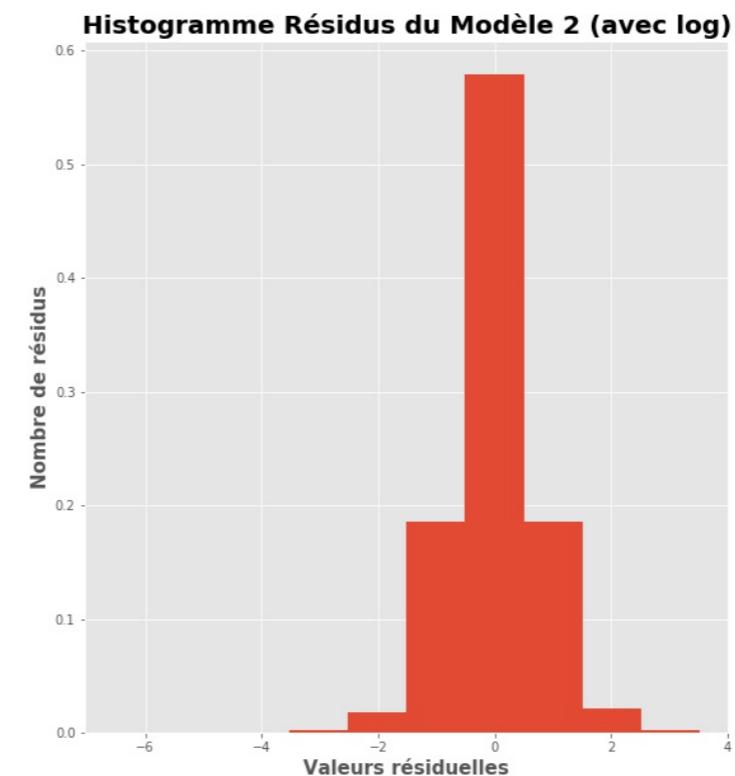
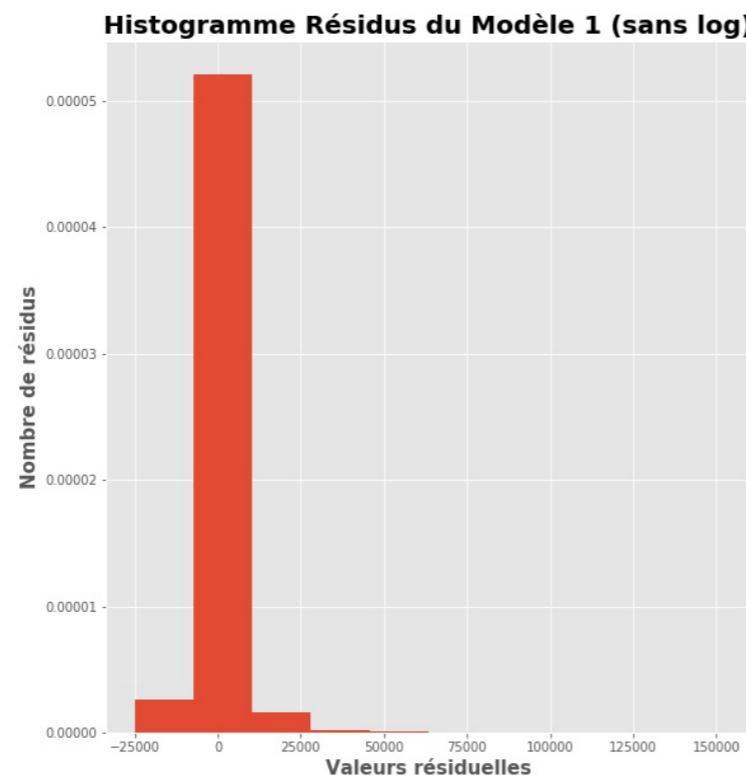
DIAGNOSTIC DES MODÈLES

ANALYSE DES RÉSIDUS : VALIDER OU INVALIDER LA RÉGRESSION

La loi normale est mieux suivie par le modèle logarithmique.

Le second modèle propose une **distribution gaussienne plus homogène**.

L'approche Q-Q Plot valide également le modèle logarithmique.



ANALYSE DES LEVIERS AVEC SEUIL

A un niveau de test 5%, le second modèle logarithmique reste toujours **plus performant avec 4.31%** (5 pays influents), contre 9.48% (11 pays influents).

Pays issus du Modèle 1 sans log.

16	Canada
17	Central African
20	Colombia
39	Guatemala
41	Honduras
43	Iceland
61	Luxembourg
77	Norway
93	South Africa
109	United Kingdom
111	United States

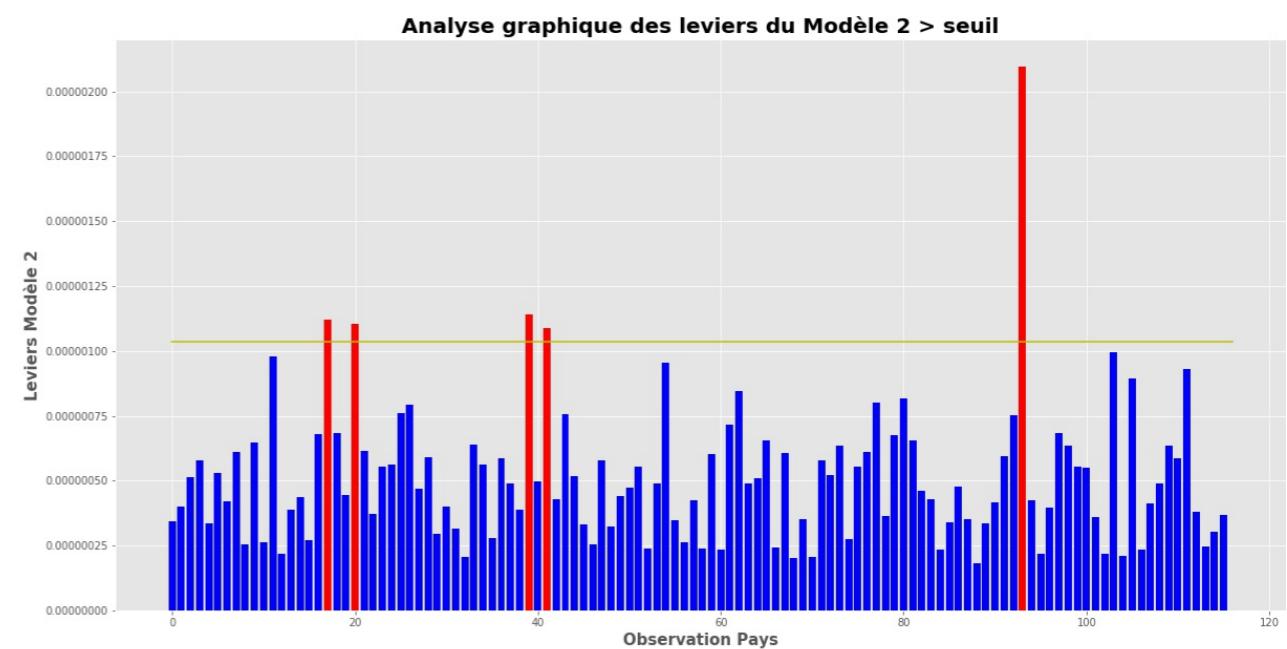
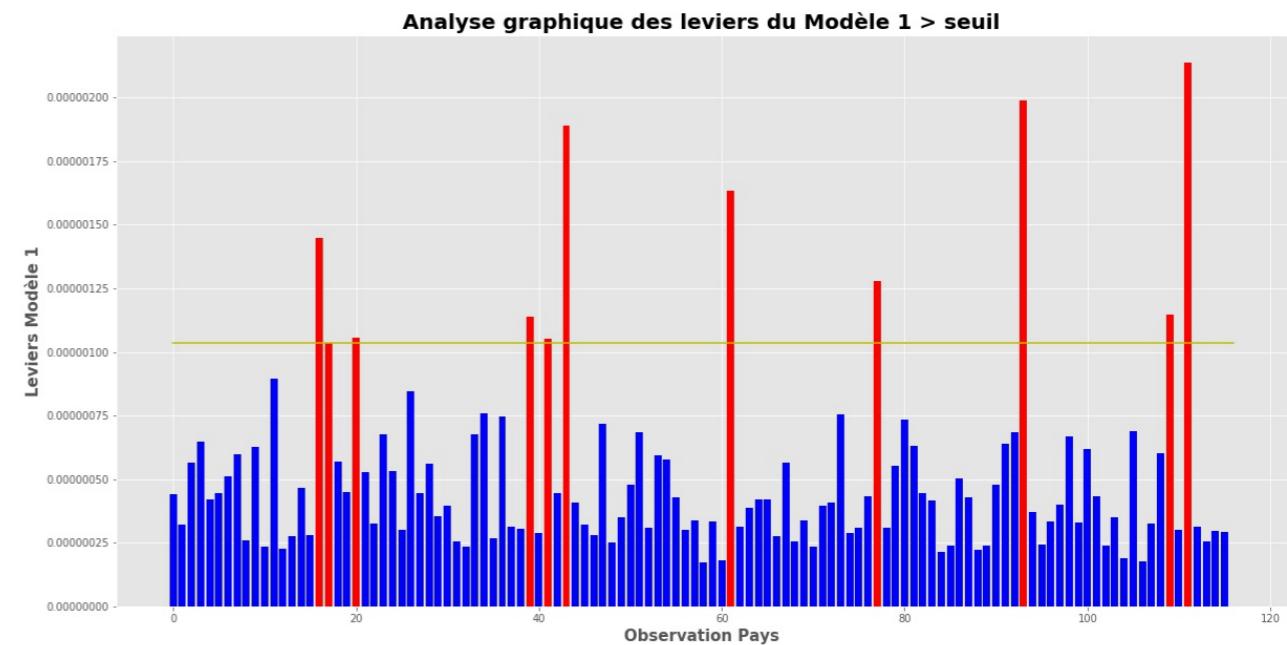
Name: country, dtype: object

Pays issus du Modèle 2 avec log.

17	Central African
20	Colombia
39	Guatemala
41	Honduras
93	South Africa

Name: country, dtype: object

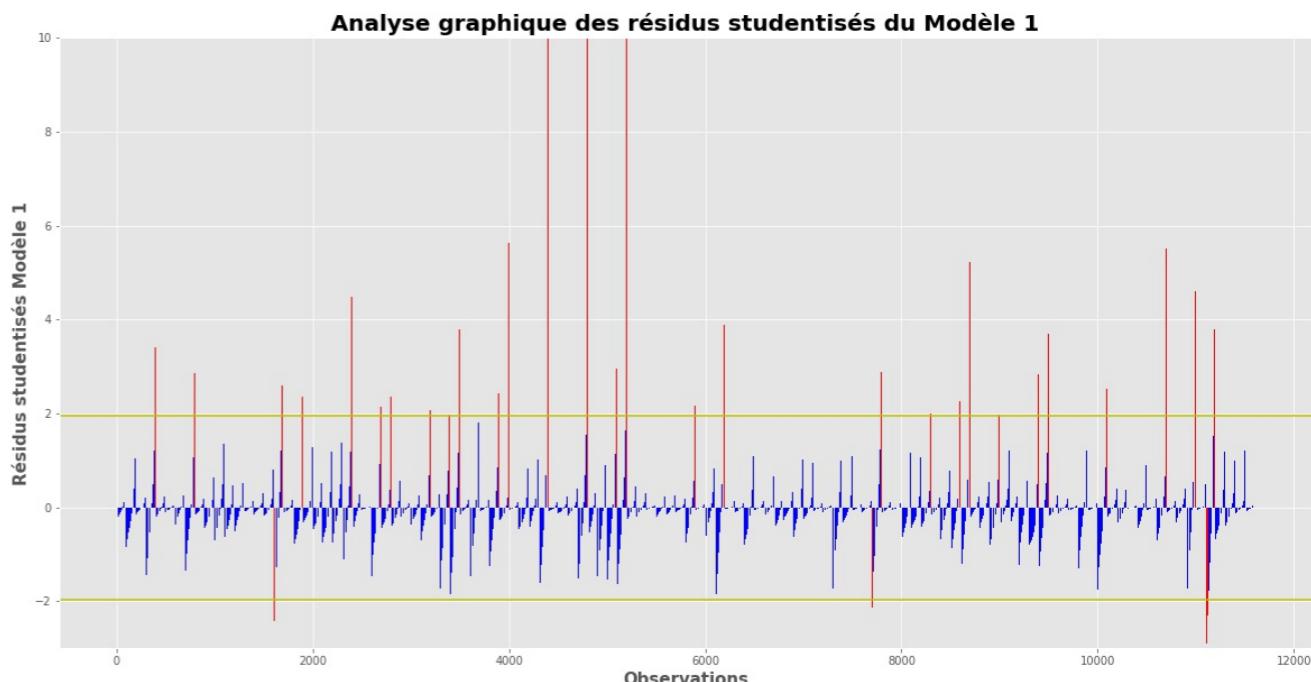
(1% le dernier modèle amélioré)



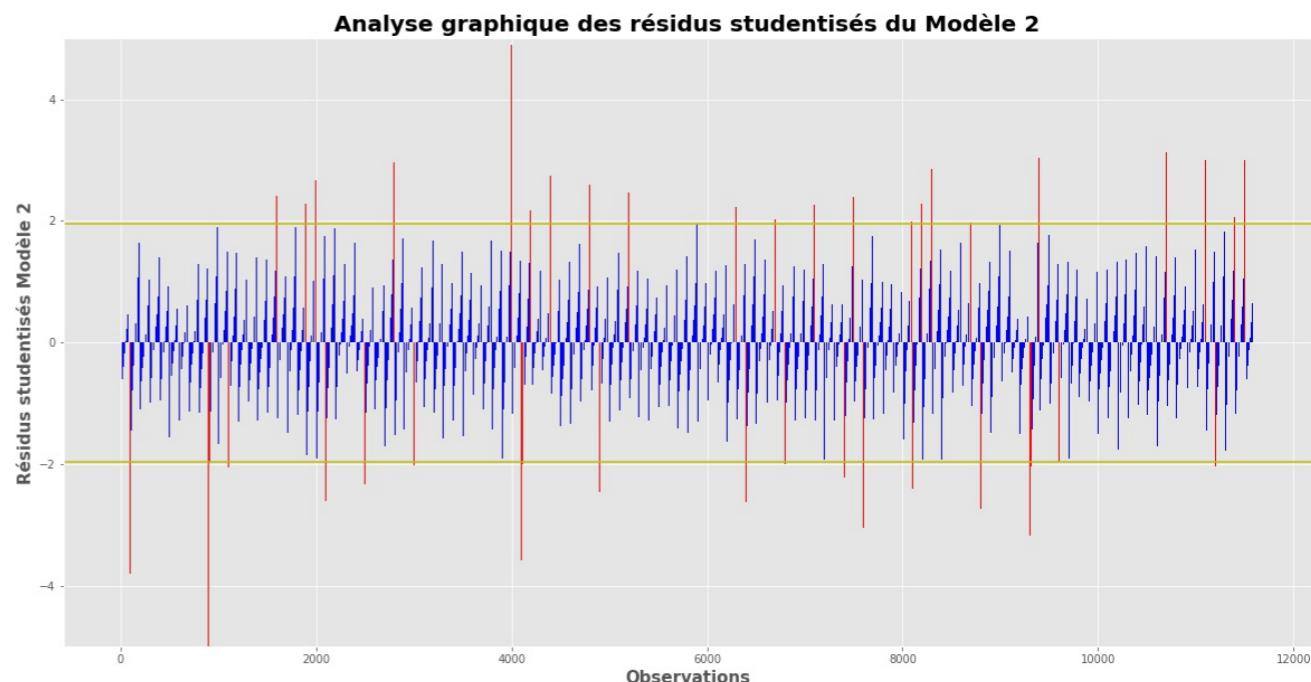
ANALYSE DES RÉSIDUS STUDENTISÉS

L'approche visuelle ne permet pas d'étudier rapidement ces résidus. L'estimation en pourcentage sera plus explicite. Ces $R_{student}$ exposent un lot de valeurs atypiques dans nos modèles.

3.53% valeurs sont hors seuil



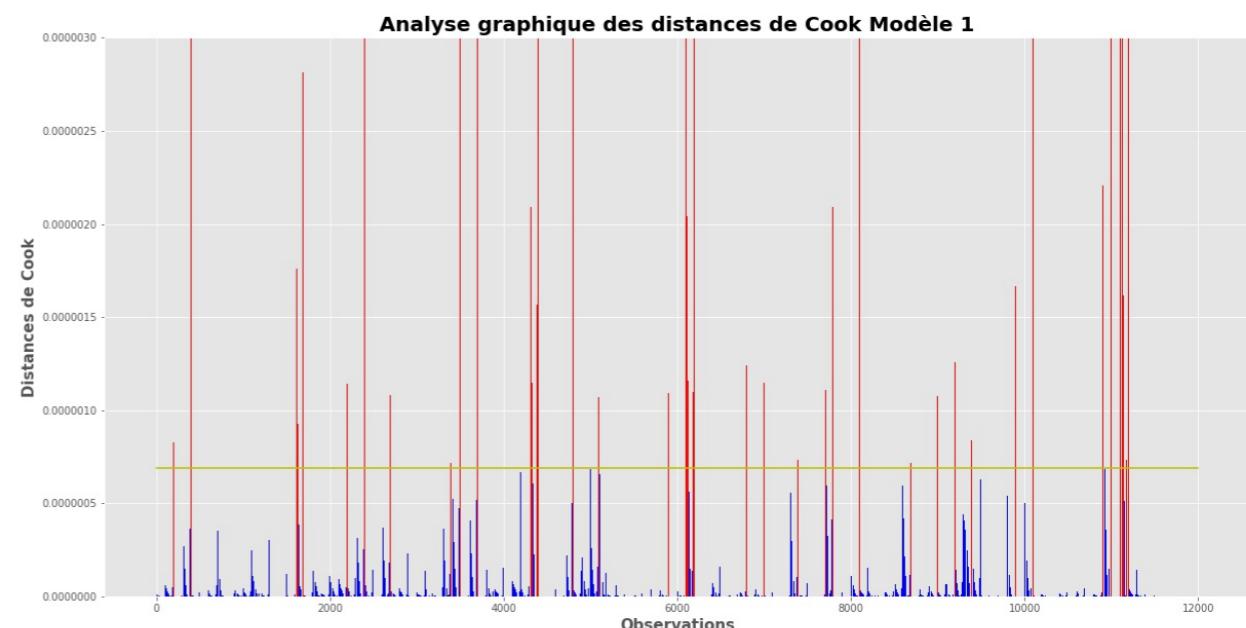
**5.47% valeurs sont hors seuil
(4% sur le dernier modèle amélioré)**



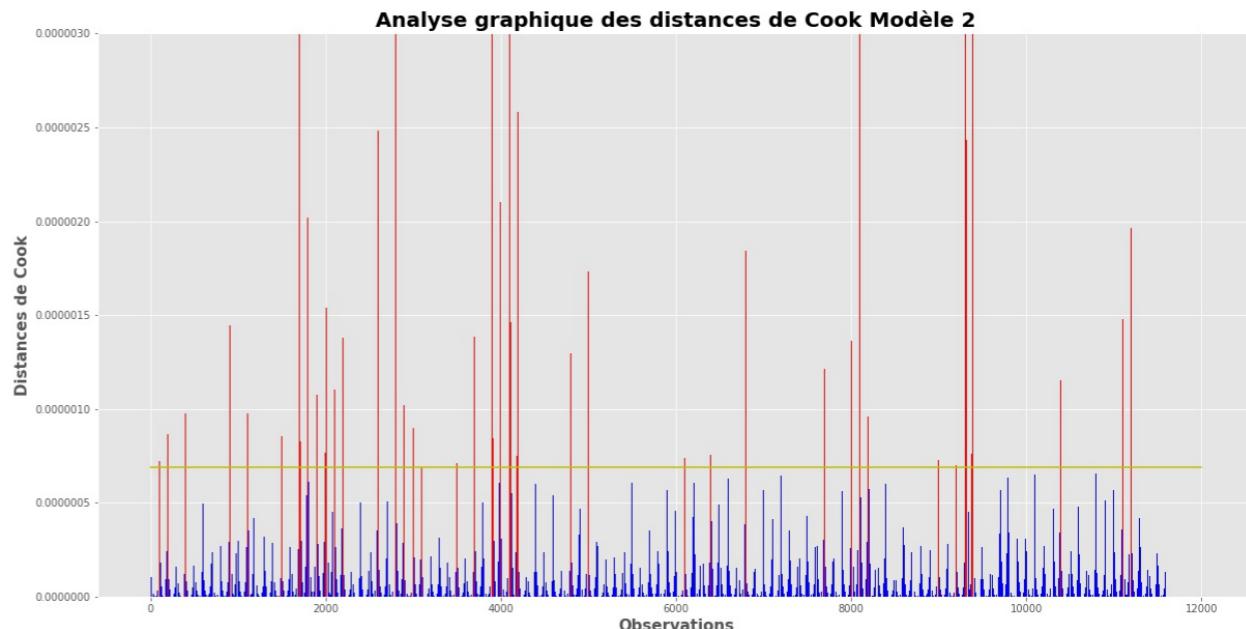
ANALYSE DE LA DISTANCE DE COOK

Ce critère permet de mieux comprendre les observations potentiellement influentes, ce qui n'est pas satisfaisant. Mesure l'influence d'une observation en prenant en compte l'effet levier et l'importance des résidus. Ces deux modèles sont plus ou moins identiques, on sait désormais que cela représente 5% des observations.

3.53% valeurs sont hors seuil



**5.47% valeurs sont hors seuil
(5% sur le dernier modèle amélioré)**



COLINÉARITÉ ET CONSTANCE DE LA VARIANCE

Ces tests statistiques ont été réalisés sur chacun des modèles de régression linéaire multiple.

Aucune colinéarité des variables : pas d'effets nocifs

```
variables = modele2_parents.model.exog  
[variance_inflation_factor(variables, i) for i in np.arange(1,variables.shape[1])]  
  
[1.0633031303104725, 1.0, 1.0633031303104825]
```

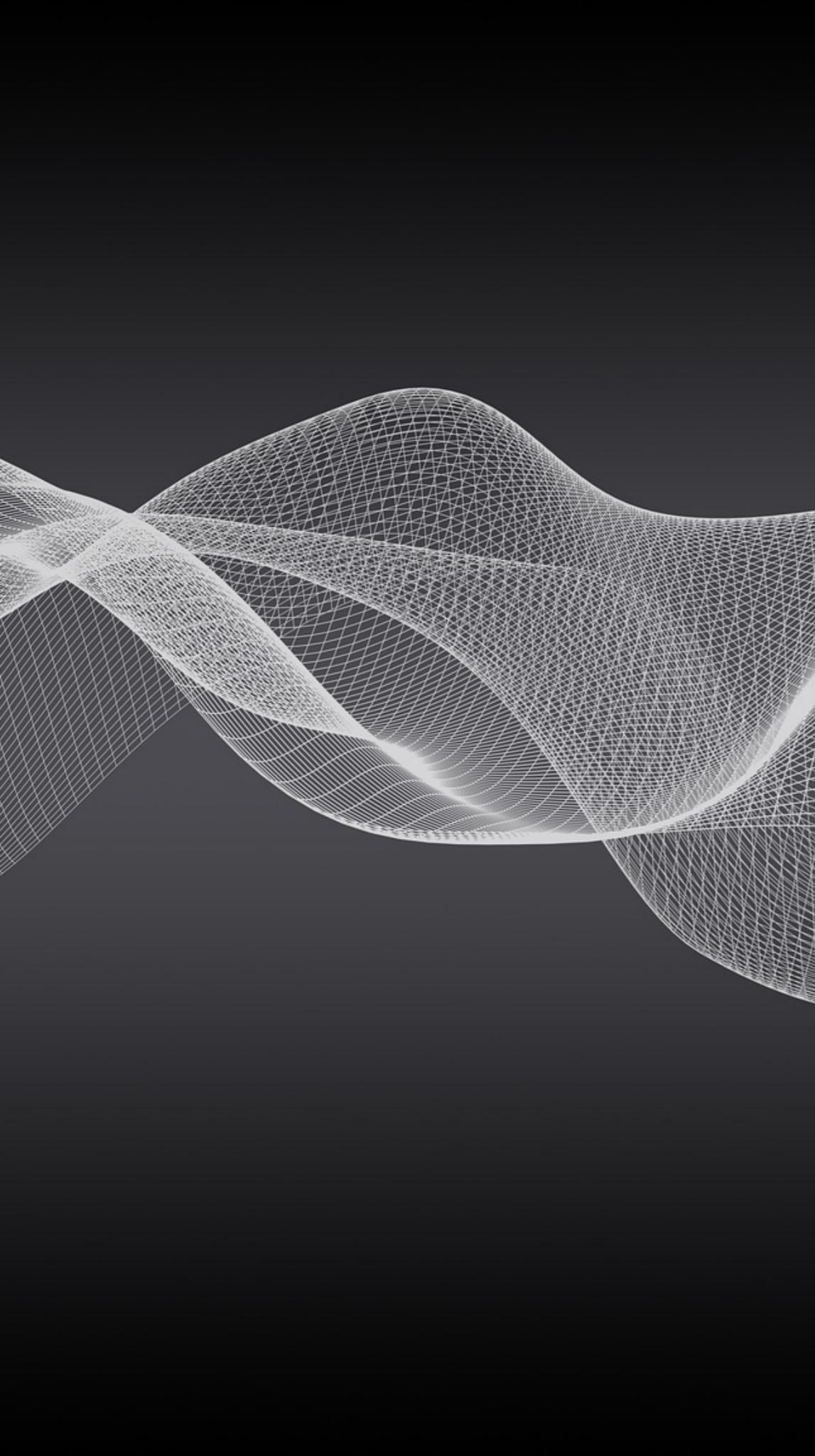
Effet de l'hétéroscédasticité, remise en question du modèle?

```
__, pval, __, f_pval = sms.het_breushpagan(modele2_parents.resid, modele2_parents.model.exog)  
print('p value test Breusch-Pagan:', pval)  
  
p value test Breusch-Pagan: 0.0
```

Rejet H0 (la constance de la variance) Hétéroscédasticité des residus

Le modèle de régression linéaire multiple est robuste pour les grands échantillons, ce qui est le cas dans notre contexte d'étude.

Le modèle est capable de supporter des écarts important aux hypothèses homoscédastique et gaussienne.

A large, abstract graphic on the left side of the slide features a complex, flowing wireframe mesh in white against a dark gray background. The mesh consists of numerous thin, intersecting lines forming a series of interconnected loops and waves, resembling a digital representation of a physical structure like a protein or a complex data visualization.

Pays de naissance : 69%
Revenu des parents : 5.5%
Gini : 1%
et autres facteurs non expliqués...

On peut noter qu'un indice de Gini élevé va engendrer des salaires plus bas dans le pays lié à une forte inégalité sur les revenus.

Pour conclure sur le dernier modèle