# 66.Lymphome

March 16, 2023

## 1 Projet Lymphome

Analyse du surcoût éventuel lié à la guérison à l'aide du Rituximab.

### 1.1 Création du dataframe des données

[1]:
```
import pandas as pd
```

[2]:
```
df0 = pd.read_csv('../data/eme2.csv')
```

Intégration des données numériques

[3]:
```
liste_nb = ['age_patient_rando', 'FOLLOW_UP'] #['ageCALC', 'annee_randoR']
```

Intégration des données binaires

[4]:
```
liste_oui_non = ['ritux', 'greffe',
                 'cv_totR2', 'cv_valvR2', 'cv_rythR2', 'cv_insufcardR2',␣
 ↪'cv_arteriopathR2', 'cv_avcR2', 'cv_thromboR2',
                 'inf_totR2', 'inf_zonaR2', 'inf_verrueR2', 'inf_hepbR2',␣
 ↪'inf_hepcR2', 'inf_tuberR2',
                 'musclos_totR2', 'musclos_rayR2', 'musclos_necroR2',␣
 ↪'musclos_fibrocouR2', 'musclos_fibroautrR2', 'musclos_arthroR2',
                 'poum_totR2', 'poum_emboR2', 'poum_pleurR2',␣
 ↪'poum_foncpulmR2', 'poum_bpocR2', 'poum_pneumoR2',
                 'k_totR2', 'autr_diabR2',
                 'bouche_totR2', 'bouche_prothR2', 'bouche_goutR2',␣
 ↪'bouche_secR2', 'bouche_compliR2',
                 'dig_totR2', 'dig_oesophR2', 'dig_occluR2', 'dig_ulcR2',␣
 ↪'dig_perfoR2',
                 'autr_insufrenR2', 'autr_trsensR2', 'autr_fatigR2',␣
 ↪'autr_deprR2', 'autr_anxR2', 'autr_suicR2']
```

[5]:
```
liste_OUI_NON = []
```

[6]:
```
liste_Yes_No = ['RELAPSE', 'TOXICITY_GR34', 'ATCD']
```

[7]:
```
liste_cols = liste_nb + liste_oui_non + liste_OUI_NON + liste_Yes_No
```

[8]:
```
df = df0[liste_cols].copy()
```

[9]:
```
liste_bin = ['sexeR', 'fatigue', 'AGE_RANDO_SUP_EG_60', 'AGE_CALC_SUP_EG_60',␣
↪'age60',
             'MEDICAMENT', 'CONSULTATION', 'HOSPITALISATION', 'TRANSPORT',␣
↪'LATE_EVENT',
             'PSY_TOT']
```

[10]:
```
for cat in liste_bin:
    les_cat = [str(k) for k in sorted(set(df0[cat].values))]
    assert(len(les_cat)==2)
    df[cat+'_'+les_cat[0].replace(' ','').replace('<','inf_')] = df0[cat].
↪apply(lambda x:x==les_cat[0])
    df[cat+'_'+les_cat[0].replace(' ','').replace('<','inf_')] =␣
↪df[cat+'_'+les_cat[0].replace(' ','').replace('<','inf_')].astype('bool')
```

[11]:
```
import numpy as np

def oui_non(x):
    if x == 'oui':
        return True
    else:
        return False

for col in liste_oui_non:
    df[col] = df[col].apply(oui_non)
    #df[col] = df[col].astype('bool')
```

[12]:
```
def OUI_NON(x):
    if x == 'OUI':
        return 1 # True
    elif x == 'NON':
        return 0 # False
    else:
        return 2 # np.NaN

for col in liste_OUI_NON:
    df[col] = df[col].apply(OUI_NON)
    df[col] = df[col].astype('bool')
```

[13]:
```
def Yes_No(x):
    if x == 'Yes':
        return 1 # True
    elif x == 'No':
        return 0 # False
    else:
        return 2 # np.NaN
```

```
for col in liste_Yes_No:
    df[col] = df[col].apply(Yes_No)
    df[col] = df[col].astype('bool')
```

Ajout des colonnes de catégories

[14]:
```
liste_categories = ['GROUPE', 'FOLLOW_UP_SUP_EG_10', 'intens', 'RELAPSE_TRT']
```

[15]:
```
for cat in liste_categories:
    for u in sorted([u for u in list(set(df0[cat].values)) if␣
 ↪isinstance(u,str)]):
        name = cat.upper()+'_'+''.join([v.capitalize() for v in u.split(' ')]).
 ↪replace('<','_inf_')
        df[name] = df0[cat].apply(lambda x:x==u)
```

Ajout du coût total

[16]:
```
target = 'COUT_TOTAL'
df[target] = df0[target].copy()
```

[17]:
```
df.head()
```

[17]:

| | age_patient_rando | FOLLOW_UP | ritux | greffe | cv_totR2 | cv_valvR2 | \ |
|---|---|---|---|---|---|---|---|
| 0 | 24.0 | 9.0 | False | False | False | False | |
| 1 | 40.0 | 9.0 | False | False | False | False | |
| 2 | 49.0 | 9.0 | False | False | False | False | |
| 3 | 43.0 | 8.0 | False | False | False | False | |
| 4 | 44.0 | 9.0 | False | False | False | False | |

| | cv_rythR2 | cv_insufcardR2 | cv_arteriopathR2 | cv_avcR2 | … | \ |
|---|---|---|---|---|---|---|
| 0 | False | False | False | False | … | |
| 1 | False | False | False | False | … | |
| 2 | False | False | False | False | … | |
| 3 | False | False | False | False | … | |
| 4 | False | False | False | False | … | |

| | RELAPSE_TRT_Chemotherapy,Radiotherapy,OtherTreatment | \ |
|---|---|---|
| 0 | False | |
| 1 | False | |
| 2 | False | |
| 3 | False | |
| 4 | False | |

| | RELAPSE_TRT_Chemotherapy,Radiotherapy,Transplant | \ |
|---|---|---|
| 0 | False | |
| 1 | False | |

```
2                                                False
3                                                False
4                                                False

   RELAPSE_TRT_Chemotherapy,Transplant  \
0                                False
1                                False
2                                False
3                                False
4                                False

   RELAPSE_TRT_Chemotherapy,Transplant,OtherTreatment  \
0                                                False
1                                                False
2                                                False
3                                                False
4                                                False

   RELAPSE_TRT_Immunotherapy  RELAPSE_TRT_OtherTreatment  \
0                      False                       False
1                      False                       False
2                      False                       False
3                      False                       False
4                      False                       False

   RELAPSE_TRT_Radiotherapy  RELAPSE_TRT_Radiotherapy,OtherTreatment  \
0                     False                                    False
1                     False                                    False
2                     False                                    False
3                     False                                    False
4                     False                                    False

   RELAPSE_TRT_Radiotherapy,Transplant  COUT_TOTAL
0                                False         0.0
1                                False         0.0
2                                False         0.0
3                                False         0.0
4                                False         0.0

[5 rows x 83 columns]
```

[19]:
```
len(df)
```

[19]: 1671

On a donc 1671 lignes pour 82 colonnes. Nettoyons les NaN…

[20]:
```
df = df.dropna()
```

[21]:
```
len(df)
```

[21]: 1670

[22]:
```
len(df[(df['COUT_TOTAL'] == 0) & (df['ritux']==True)]),␣
↪len(df[(df['COUT_TOTAL'] == 0) & (df['ritux']==False)])
```

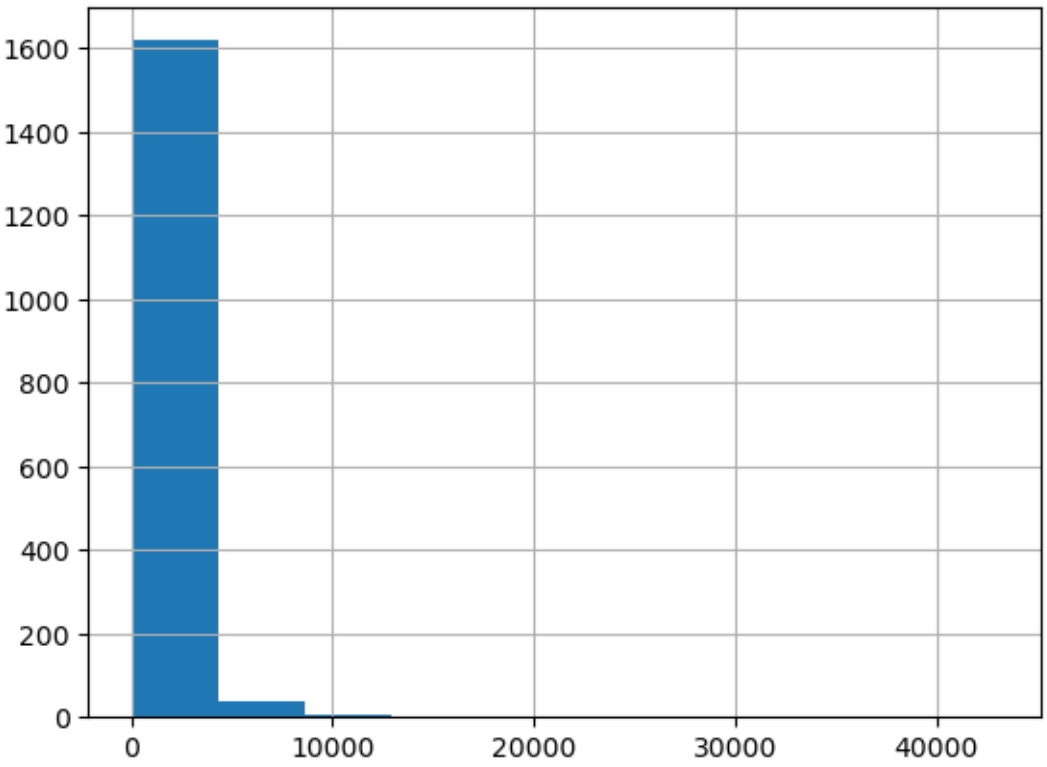
[22]: (273, 297)

273 individus ayant pris du Rituximab ont un coût nul, pour 297 n'en ayant pas pris. Il faudrait peut-être tester la significance de cette différence, qui apparaît cependant assez fortuite (on ne s'attendait pas à une égalité stricte).

Histogramme des coûts totaux...

[23]:
```
df.COUT_TOTAL.hist()
```

[23]: <AxesSubplot: >




Certains coûts sont bien plus élevés que d'autres, ce qui rend la comparaison impossible. D'autre part, l'absence de soin sur une longue période semble étrange. On décide donc (dans un premier

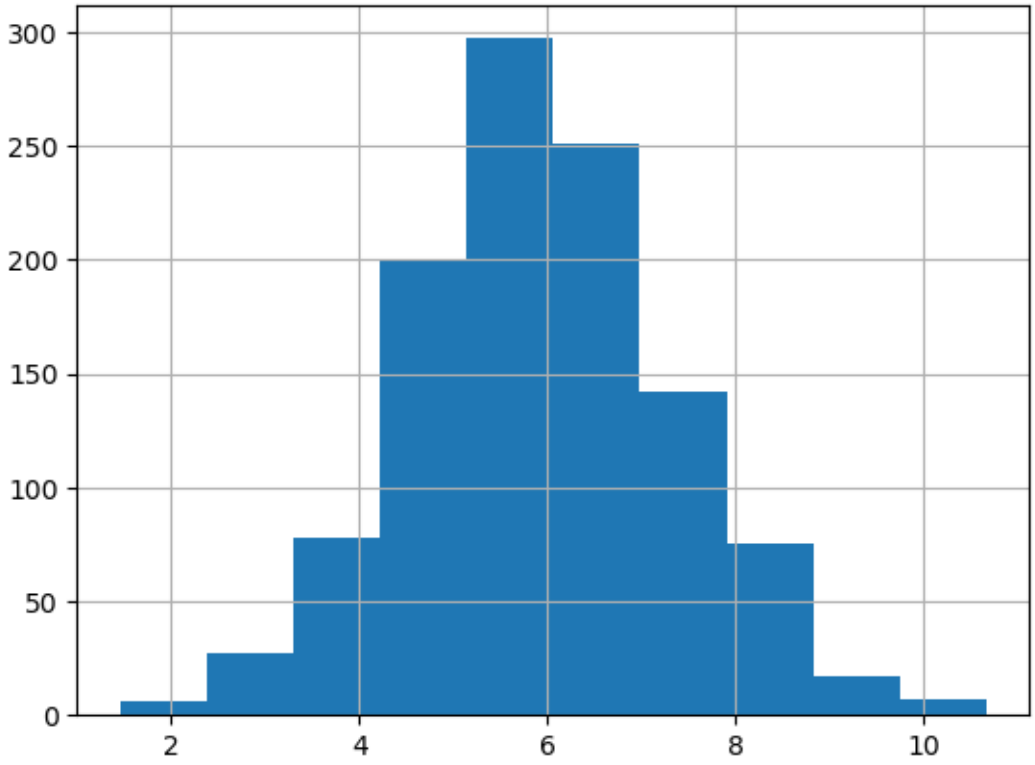
temps ?) d'écarter les coûts nuls, et de passer au log.

[24]:
```
df.to_pickle('Nerich_dataframe.pkl')
df.to_csv('Nerich_dataframe.csv')

df = df[df['COUT_TOTAL'] > 0]
df['COUT_TOTAL'] = np.log(df['COUT_TOTAL'])
```

Nouvel histogramme :

[25]:
```
df.COUT_TOTAL.hist()
```

[25]: <AxesSubplot: >




Sauvegarde des données...

[26]:
```
df.to_pickle('Nerich_dataframe_logpos.pkl')
df.to_csv('Nerich_dataframe_logpos.csv')
```

## 1.2 Premières visualisations

### 1.2.1 Statistiques descriptives

Moyenne, écart-type, étendue et quartiles du coût total, en prenant ou non en compte le médicament.

[27]:
```
df.COUT_TOTAL.describe()
```

[27]:
```
count    1100.000000
mean        5.934557
std         1.410408
min         1.457063
25%         4.997212
50%         5.914609
75%         6.813448
max        10.670209
Name: COUT_TOTAL, dtype: float64
```

[28]:
```
df.loc[df['ritux']].COUT_TOTAL.describe()
```

[28]:
```
count    556.000000
mean       5.869761
std        1.449989
min        1.457063
25%        4.848908
50%        5.825303
75%        6.758086
max       10.670209
Name: COUT_TOTAL, dtype: float64
```

[29]:
```
df.loc[df['ritux']==False].COUT_TOTAL.describe()
```

[29]:
```
count    544.000000
mean       6.000781
std        1.366932
min        1.488964
25%        5.186574
50%        5.983728
75%        6.850941
max       10.539221
Name: COUT_TOTAL, dtype: float64
```

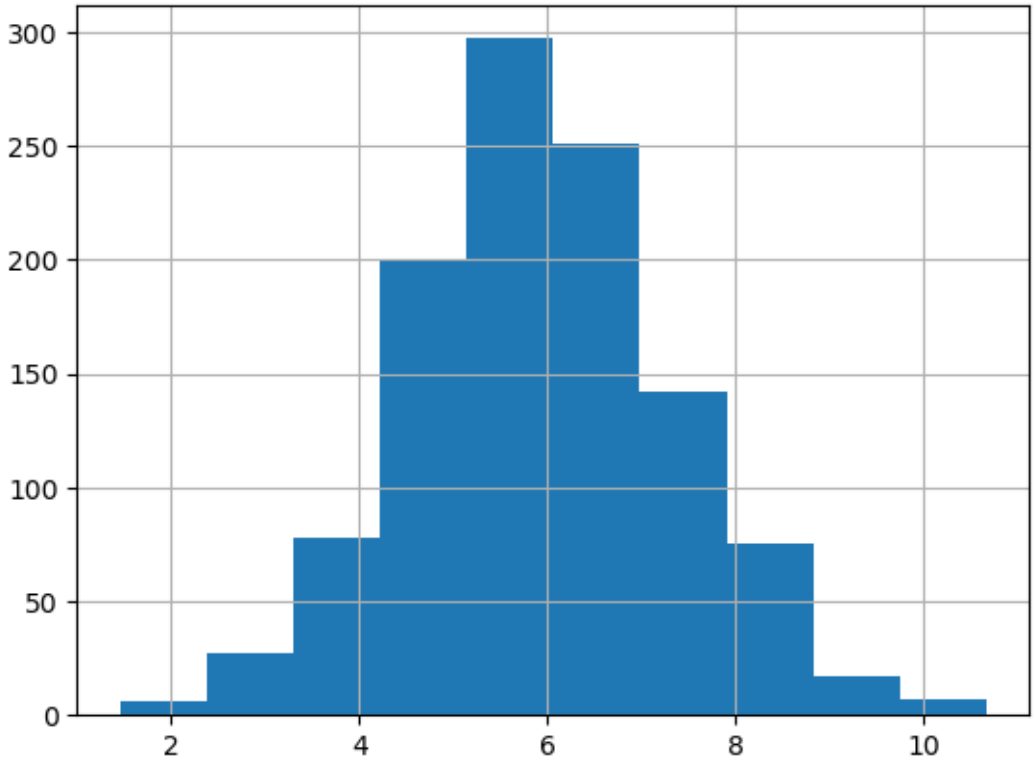
Le coût moyen des indivudus ayant été traités avec le Rituximab (5.869761) est donc très légèrement inférieur à celui de ceux n'ayant pas été traités avec (6.000781). Mais la différence ne semble pas significative : à tester. Idem pour les autres statistiques : a priori, pas de différence.

### 1.2.2 Histogrammes des coûts

Histogrammes du coût total, coût total après prise de Rituximab, et coût total sans prise.

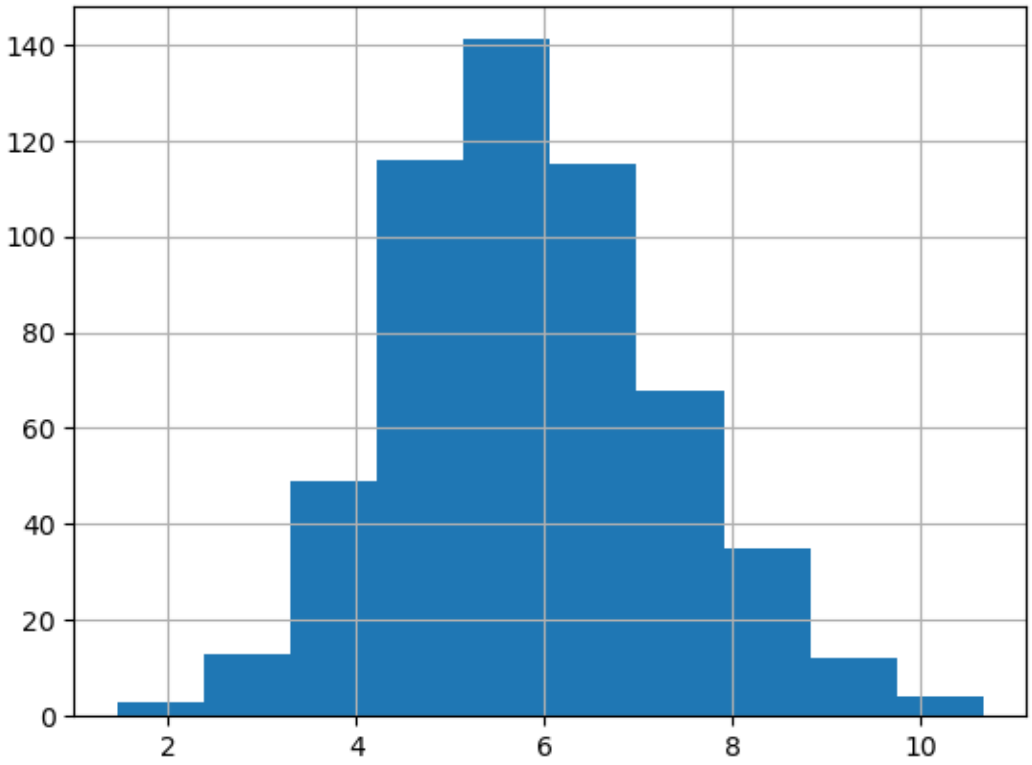
[30]:
```
df.COUT_TOTAL.hist()
```

[30]: <AxesSubplot: >



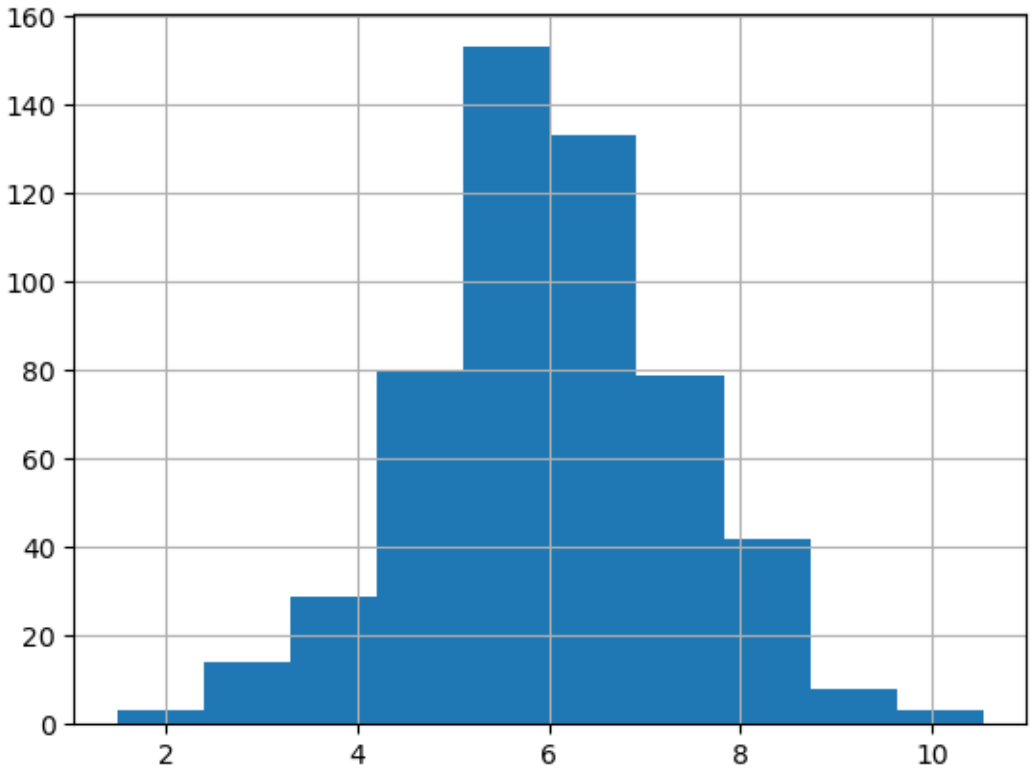

[31]:
```
df.loc[df['ritux']].COUT_TOTAL.hist()
```

[31]: <AxesSubplot: >




[32]:
```
df.loc[df['ritux']==False].COUT_TOTAL.hist()
```

[32]: <AxesSubplot: >

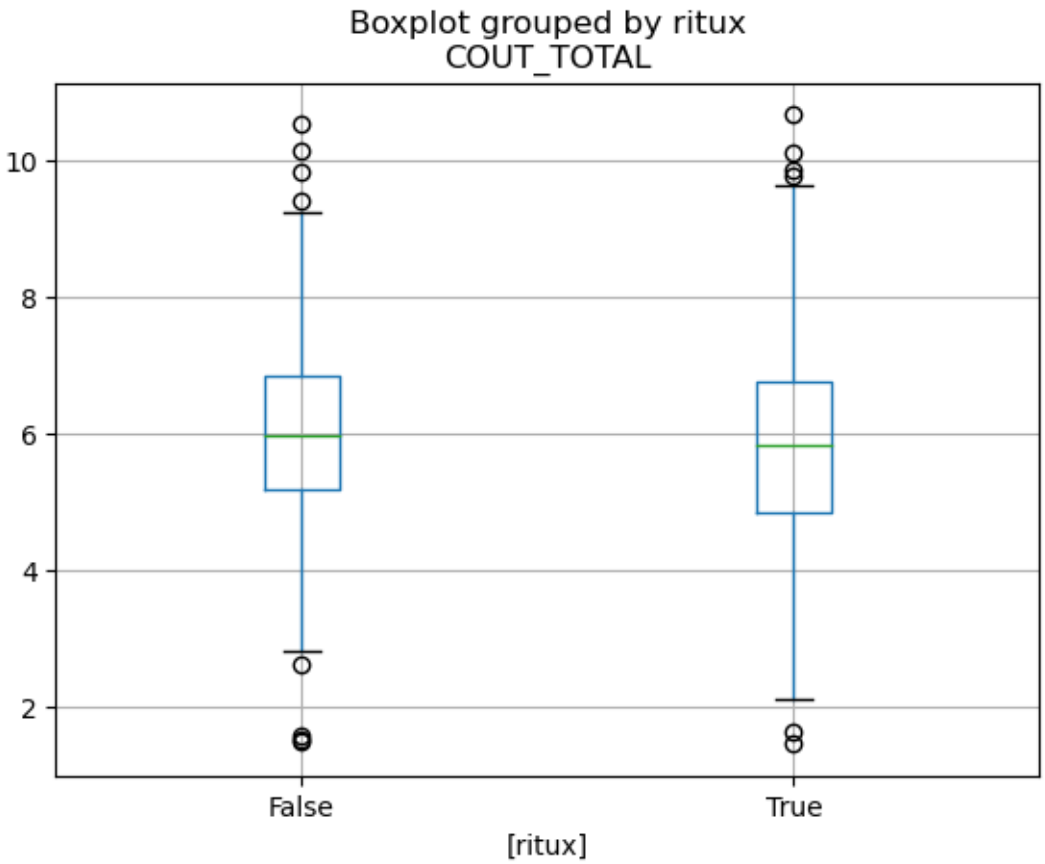



### 1.2.3 Boxplot, violinplot

Version boîte à moustache du coût total, avec ou sans Rituximab.

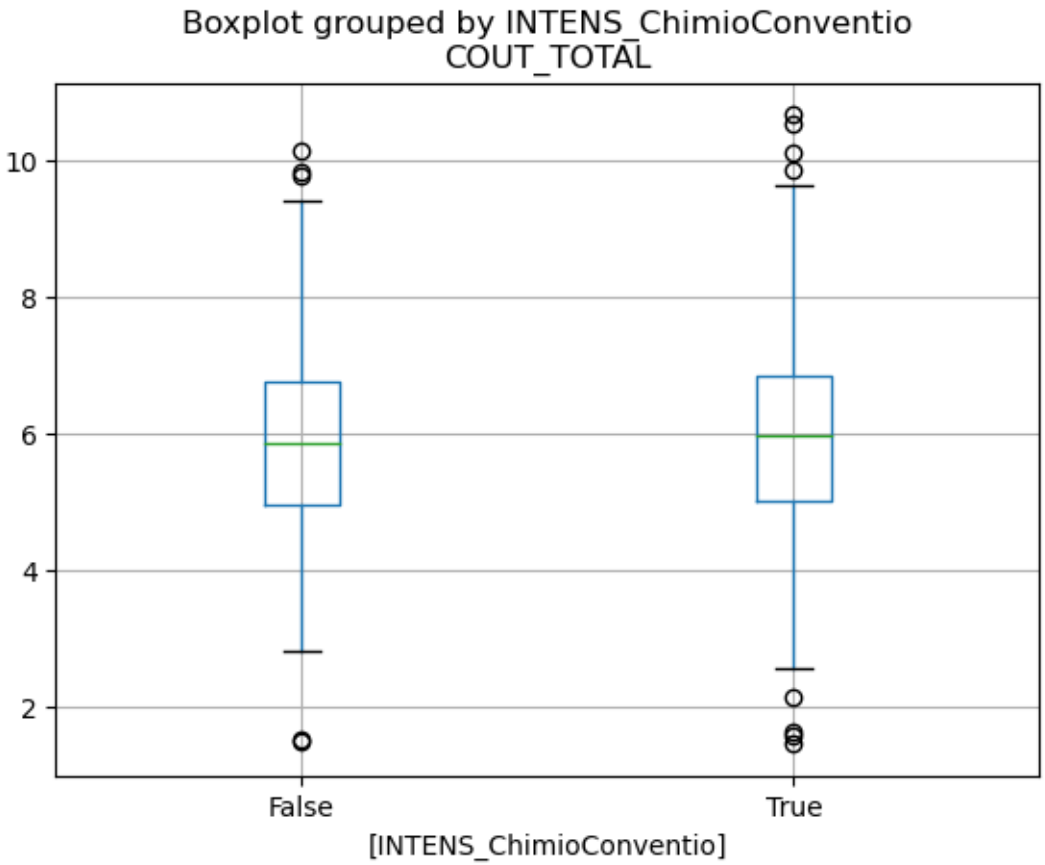
[33]:
```
df[['ritux', 'COUT_TOTAL']].boxplot(by='ritux')
```

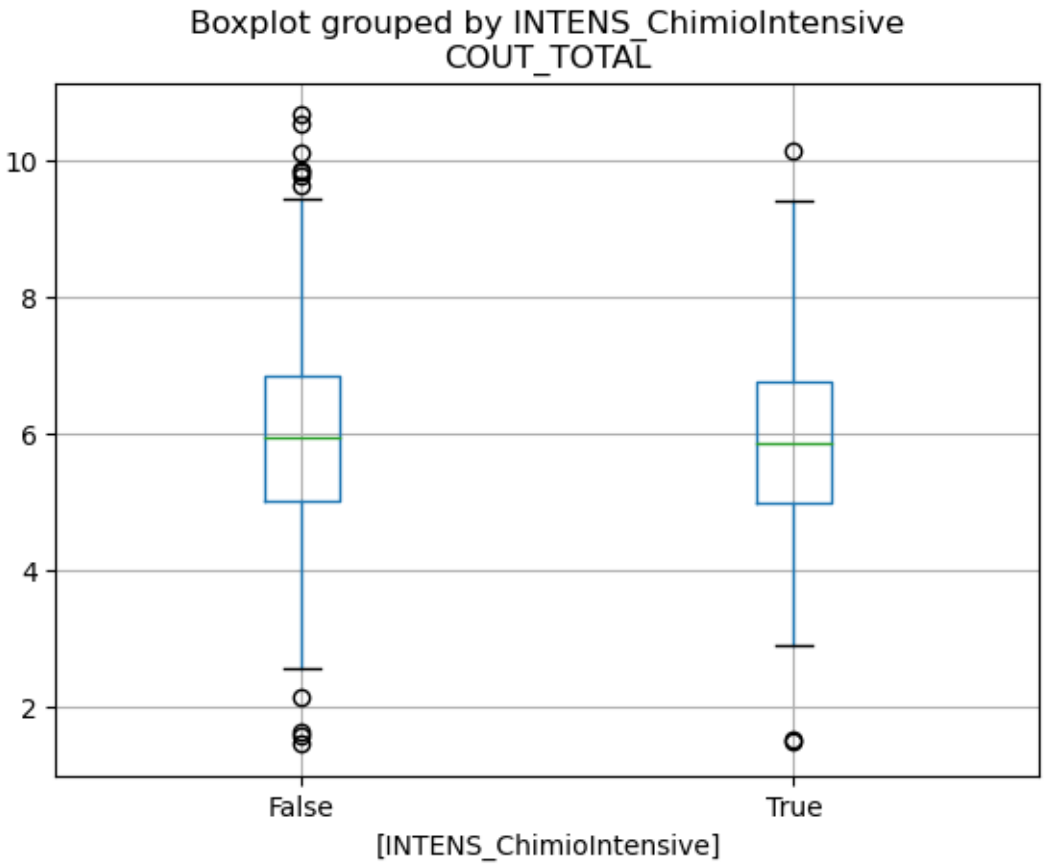
[33]:
```
<AxesSubplot: title={'center': 'COUT_TOTAL'}, xlabel='[ritux]'>
```

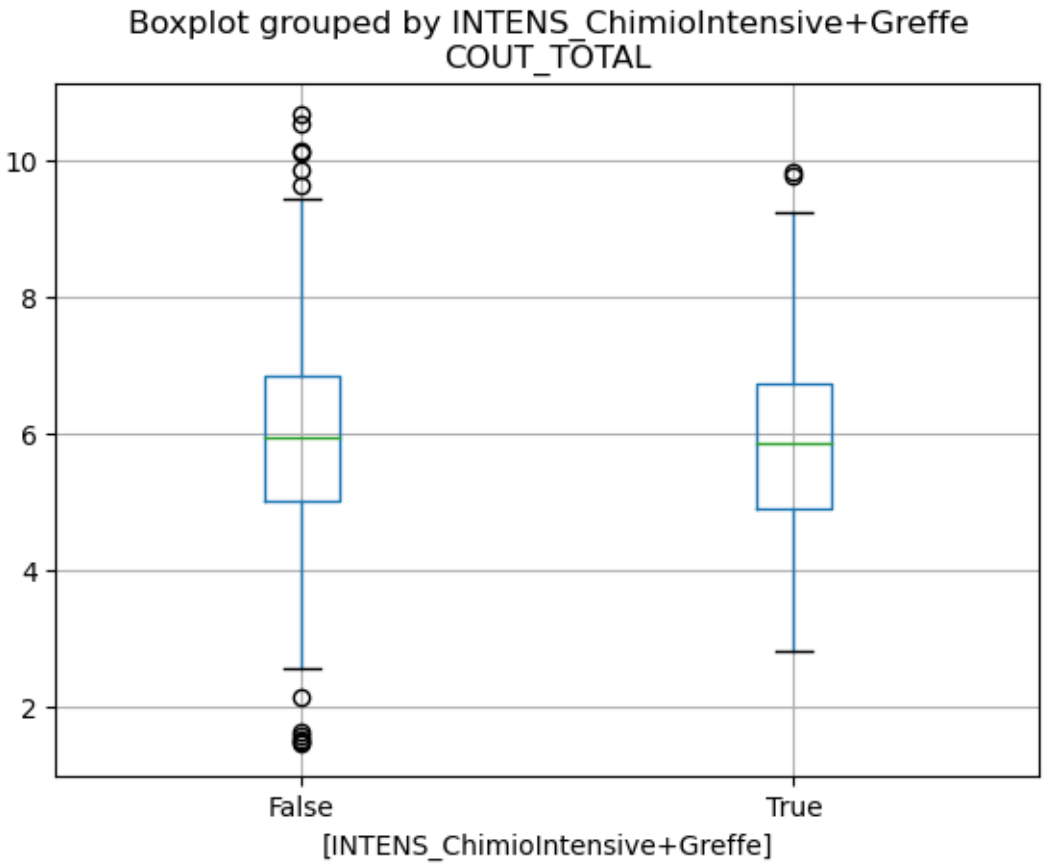



Idem, pour les chimiothérapies.

[34]:
```
for intens in ['INTENS_ChimioConventio', 'INTENS_ChimioIntensive',␣
↪'INTENS_ChimioIntensive+Greffe']:
    df[[intens, 'COUT_TOTAL']].boxplot(by=intens)
```


Boxplot grouped by INTENS_ChimioConventio
COUT_TOTAL
10
8
6
4
2
False
True
[INTENS_ChimioConventio]

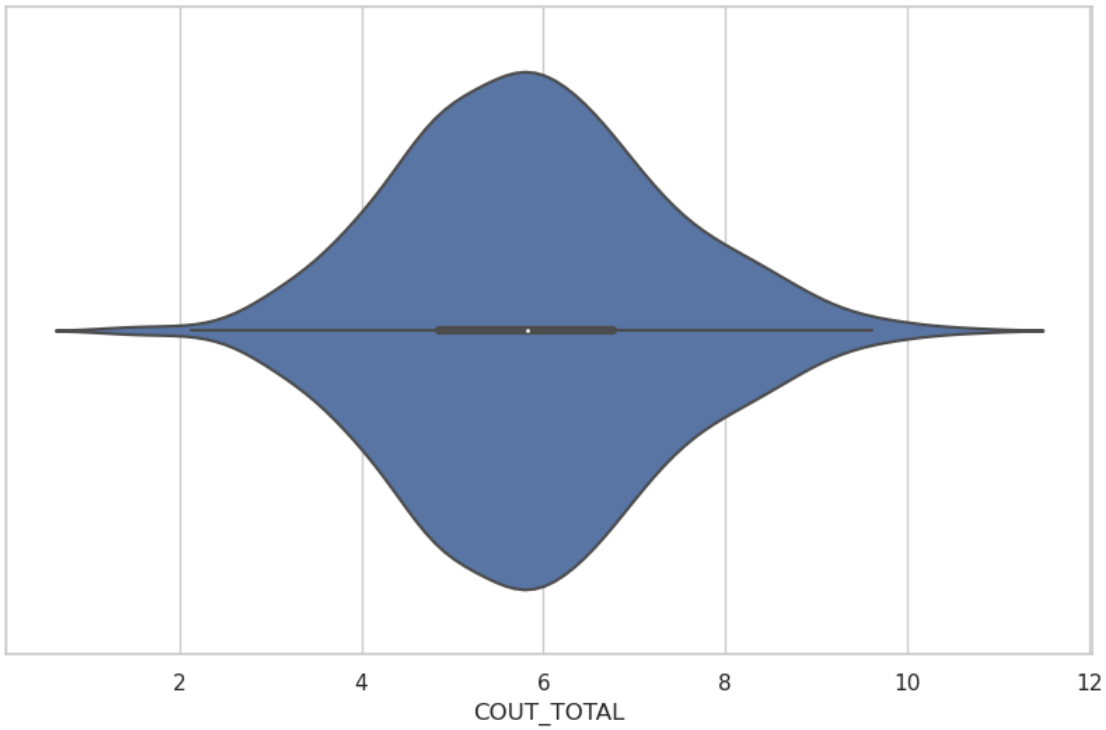

Boxplot grouped by INTENS_ChimioIntensive
COUT_TOTAL
10
8
6
4
2
False
True
[INTENS_ChimioIntensive]



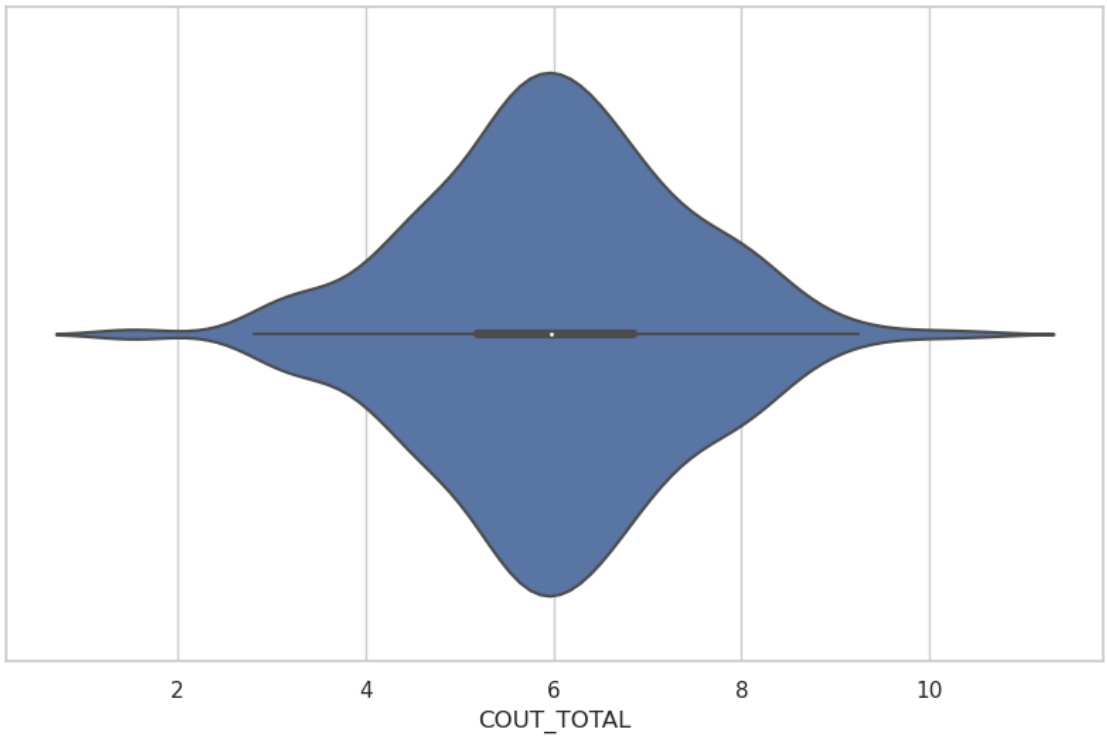

A l'oeil, les différences n'ont pas l'air significatives. Version violinplot...

[35]:
```
%pylab inline
pylab.rcParams['figure.figsize'] = (10, 6)
import seaborn as sns
sns.set(style="whitegrid")
ax = sns.violinplot(x=df.loc[df['ritux']].COUT_TOTAL)
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```




[36]:
```
ax = sns.violinplot(x=df.loc[df['ritux']==False].COUT_TOTAL)
```



### 1.2.4 Intervalle de confiance des moyennes

On construit un intervalle de confiance de niveau 95% pour la moyenne de chaque distribution, à partir des échantillons que l'on a, pour voir dans quelle mesure ces derniers se chevauchent :

[37]:
```
import statsmodels.stats.api as sms
sms.DescrStatsW(df.loc[df['ritux']].COUT_TOTAL).tconfint_mean()
```

[37]:
```
(5.748973581840167, 5.990549377522008)
```

La confiance que l'on a pour que cet intervalle contienne la moyenne (réelle) des coûts totaux après prise de Rituximab est donc de 95%. On calcule le même pour le coût sans prise de Rituximab durant le traitement du lymphome :

[38]:
```
sms.DescrStatsW(df.loc[df['ritux']==False].COUT_TOTAL).tconfint_mean()
```

[38]:
```
(5.885657745460199, 6.115905123391014)
```

Le chevauchement est raisonnable.

## 1.3 Comparaison de deux moyennes

Nous avons deux populations, à savoir les individus traités par Rituximab et ceux traités sans, pour lesquels on a les coûts globaux des traitements post-guérison de deux échantillons. Notre variable explicative est binaire, celle à expliquer est numérique, conduisant à deux moyennes pour chacun des deux échantillons.

On veut tester l'égalité des moyennes $\mu_1$ et $\mu_2$, inconnues, des deux populations, au regard des données collectées (celles des deux échantillons). Nous testons donc l'hypothèse $H_0 : \mu_1 = \mu_2$ contre l'hypothèse alternative $H_1 : \mu_1 \neq \mu_2$.

On compte le faire via un test t de Student à deux échantillons indépendants, qui présuppose la normalité et l'égalité des variances des deux échantillons. Les étapes de cette étude sont donc les suivantes : 1. tester la normalité des échantillons, 2. tester l'homogénéité des variances (condition dite d'homoscédasticité), 3. tester l'égalité des moyennes.

### 1.3.1 Tests de normalité

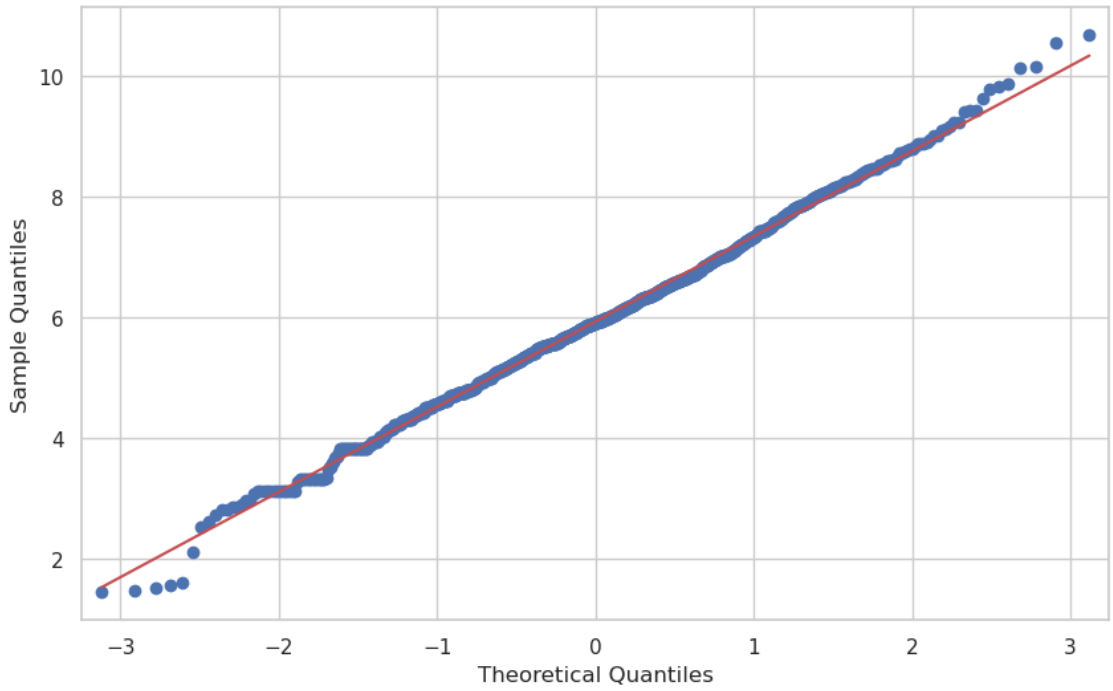
On regarde si le log du coût total, restreint à la prise ou non du Rituximab, peut être considéré comme normal, tout d'abord d'une manière graphique, pour s'en convaincre à l'oeil, puis via le test de Shapiro-Wilk.

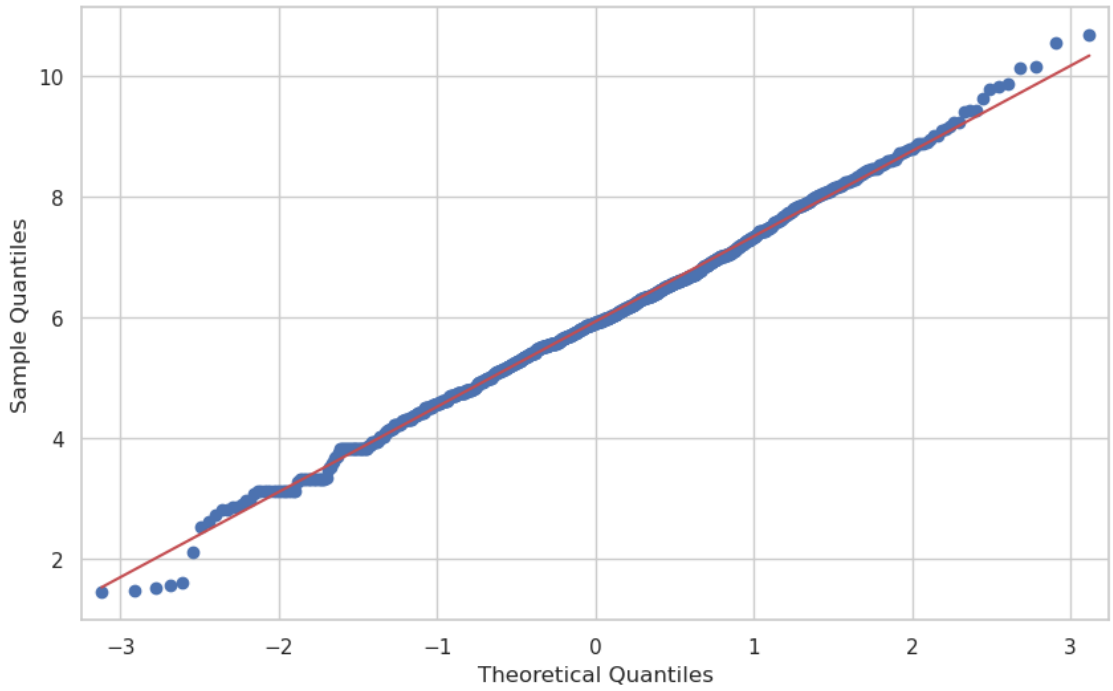
#### QQ-plot

[39]:
```
from matplotlib import pyplot
import statsmodels.api as sm
```

[40]:
```
sm.qqplot(df['COUT_TOTAL'], line='s')
```

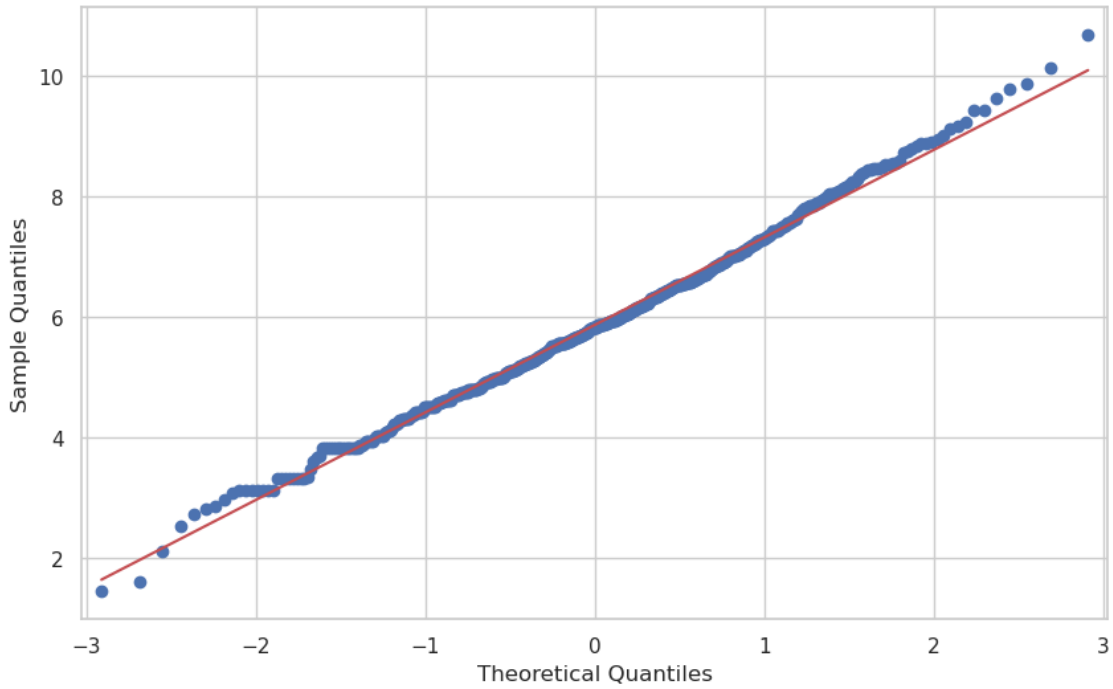
[40]:

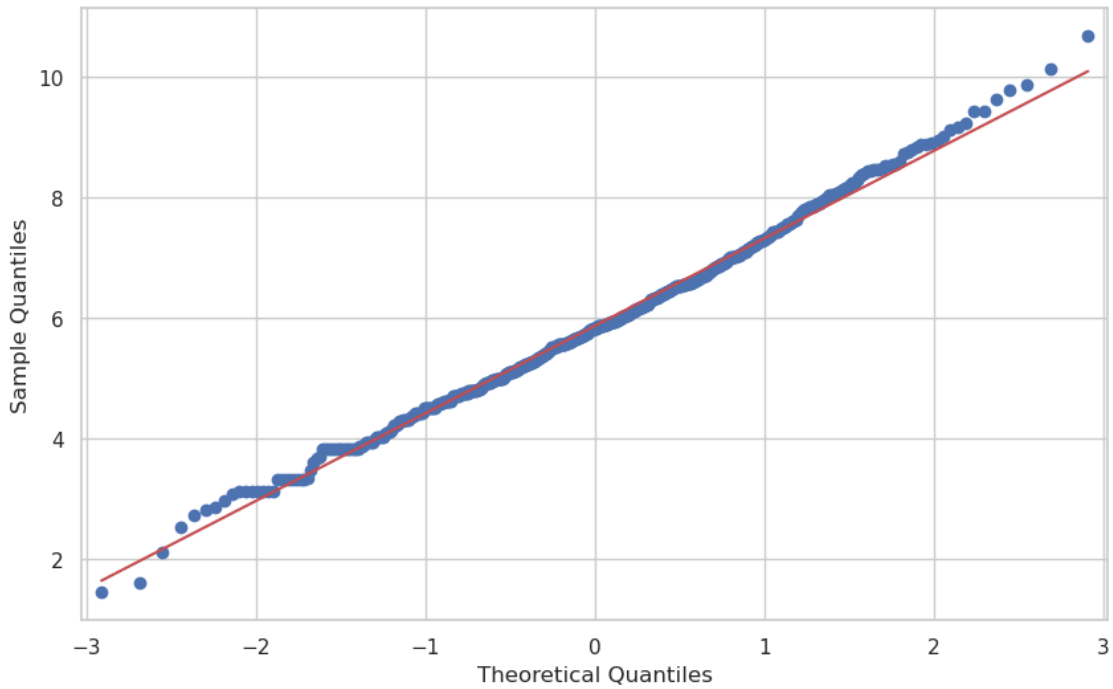



[41]:
```
sm.qqplot(df.loc[df['ritux']].COUT_TOTAL, line='s')
```

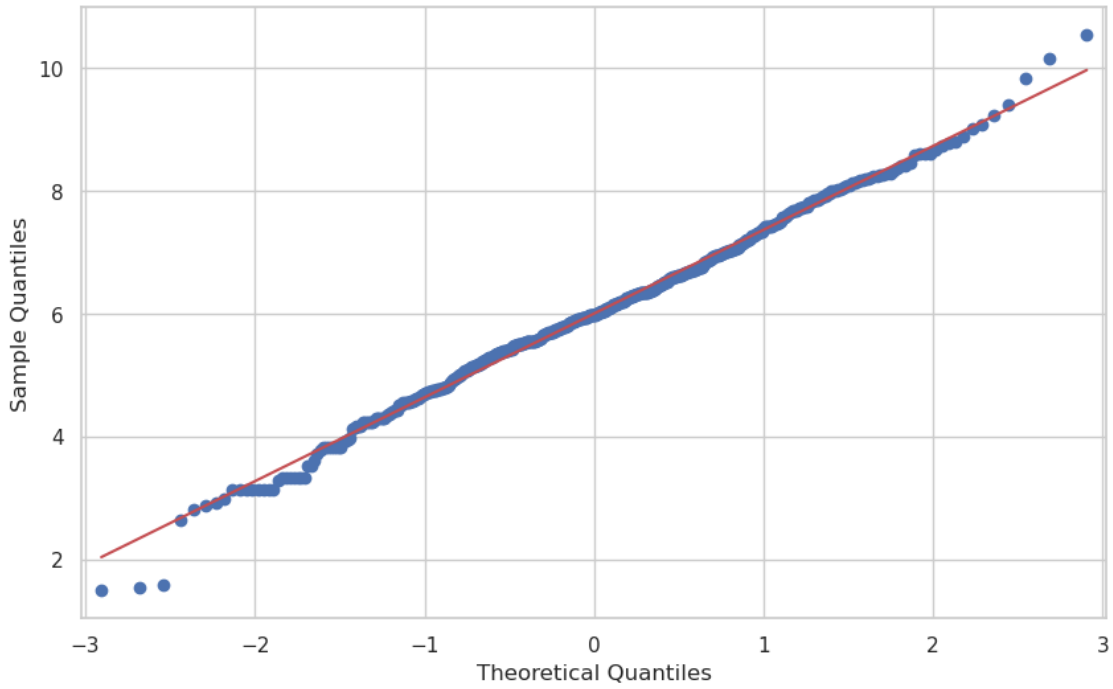
[41]:

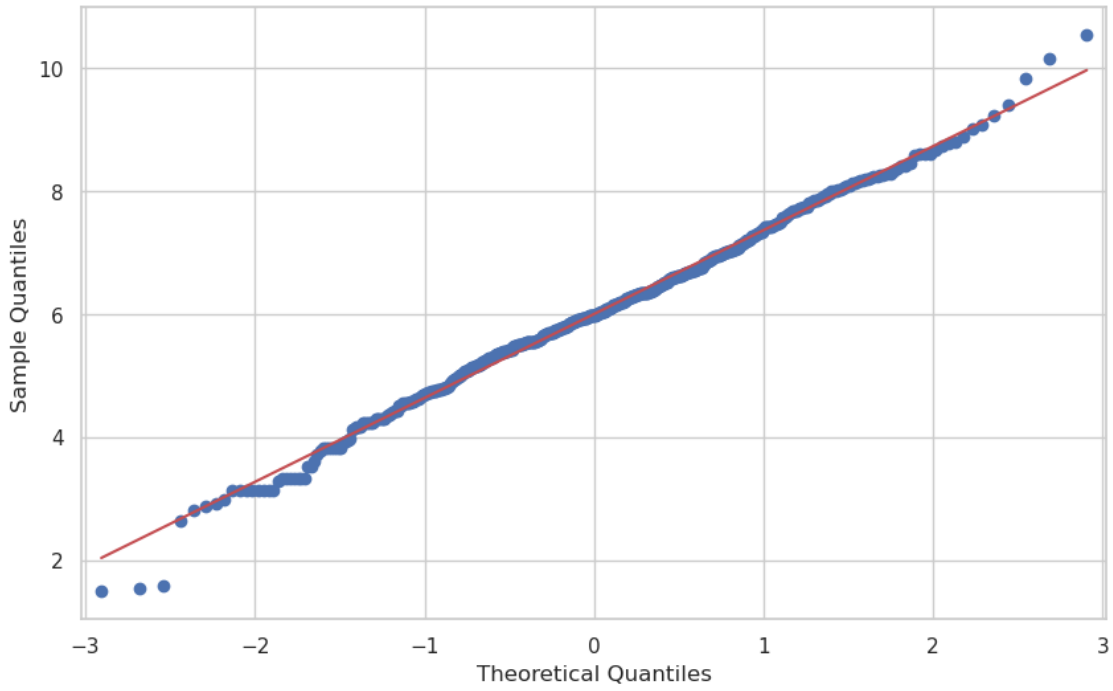



[42]:
```
sm.qqplot(df.loc[df['ritux']==False].COUT_TOTAL, line='s')
```

[42]:




Le diagramme Quantile-Quantile de comparaison d'une distribution observée avec une loi gaussienne réduite tend à nous faire accepter visuellement l'hypothèse de normalité. Il y a bien quelques

petites déviations, surtout au bas de la parcelle, mais c'était prévisible étant donné le petit échantillon de données.

**Test de Shapiro-Wilk** L'hypothèse nulle est que la population est normalement distribuée. - si la p-value est inférieure à un niveau alpha choisi (par exemple 0.05), alors l'hypothèse nulle est rejetée (i.e. il est improbable d'obtenir de telles données en supposant qu'elles soient normalement distribuées). - si la p-value est supérieure au niveau alpha choisi (par exemple 0.05), alors on ne doit pas rejeter l'hypothèse nulle. La valeur de la p-value alors obtenue ne présuppose en rien de la nature de la distribution des données.

[43]:
```
from scipy.stats import shapiro
stat, p = shapiro(df.loc[df['ritux']].COUT_TOTAL)
print('Statistics=%.3f, p=%.3f' % (stat, p))
stat, p = shapiro(df.loc[df['ritux']==False].COUT_TOTAL)
print('Statistics=%.3f, p=%.3f' % (stat, p))
```

```
Statistics=0.996, p=0.152
Statistics=0.996, p=0.144
```

On ne peut donc pas rejeter l'hypothèse H0, à savoir que ces populations sont normalement distribuées.

NB: La taille des échantillons étant suffisamment grande (n >30), on aurait pu ignorer le test de normalité sans problème majeur.

### 1.3.2 Test d'homogénéité des variances

Le test t de Student à deux échantillons non-appariés suppose l'homogénéité des variances des deux groupes à comparer, que l'on peut tester avec Bartlett, ou Levene, ce dernier étant plus robuste en cas de non normalité :

[44]:
```
import scipy
X = df.loc[df['ritux']==True].COUT_TOTAL
Y = df.loc[df['ritux']==False].COUT_TOTAL
scipy.stats.bartlett(X,Y)
```

[44]:
```
BartlettResult(statistic=1.906324226734505, pvalue=0.16737214178105853)
```

[45]:
```
scipy.stats.levene(X,Y)
```

[45]:
```
LeveneResult(statistic=2.660075899789887, pvalue=0.10318260150004634)
```

A chaque fois, des p-values > 0.05 : on ne peut pas rejeter l'hypothèse d'homoscédascticité. Les variances des deux groupes ne diffèrent pas significativement, et il y a homogénéité des variances : on peut donc se lancer dans le T-test.

### 1.3.3 Test final d'égalité des moyennes

On est donc dans le cadre d'application du test de Student pour échantillons indépendants. Nous allons tester notre hypothèse nulle que les deux échantillons sont issus de populations ayant la

même moyenne. Les groupes, ici, étant ceux ayant pris, ou pas, du Rituximab.

[46]:
```
scipy.stats.ttest_ind(X, Y)
```

[46]:
```
Ttest_indResult(statistic=-1.5413626483365679, pvalue=0.12351668150856324)
```

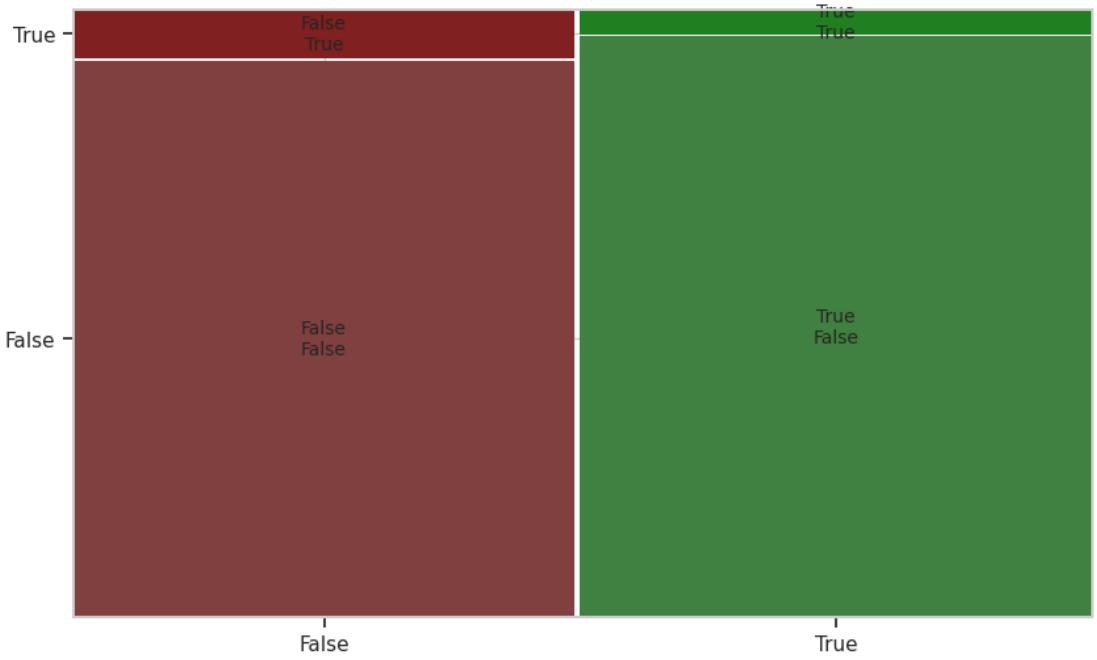
La p-value étant $> 0.05$, notre hypothèse d'égalité des moyennes ne peut être rejetée. Les moyennes ne diffèrent donc pas significativement entre elles, il n'y a pas d'effet du Rituximab sur le coût total.

### 1.4 Test $\chi^2$ d'indépendance, facteurs dichotomiques

[79]:
```
from scipy.stats import chi2_contingency
dg = pd.crosstab(df['ritux'], df['cv_insufcardR2'])
print(dg)
X2value, pvalue, degree_of_freedom, expected = chi2_contingency(dg)
print("Effectif théorique :\n")
print(pd.DataFrame(data=expected[:,:],
             index=dg.index,
             columns=dg.columns).round(2))
print(f"\np-value : {pvalue:.4f}")
```

```
cv_insufcardR2  False  True
ritux
False             500     44
True              533     23
Effectif théorique :

cv_insufcardR2   False  True
ritux
False           510.87  33.13
True            522.13  33.87

p-value : 0.0090
```

[49]:
```
from statsmodels.graphics.mosaicplot import mosaic
mosaic(df, ['ritux', 'cv_insufcardR2']);
```




[50]:
```
from statsmodels.graphics.mosaicplot import mosaic
from scipy.stats import chi2_contingency

pylab.rcParams['figure.figsize'] = (8, 5)

def teste_independance(facteur):
    print(f'\n\n - cas de {facteur} :\n')
    df2 = pd.crosstab(df['ritux'],df[facteur])
    print("Tableau de contingence :\n")
    print(df2)
    X2value, pvalue, degree_of_freedom, expected = chi2_contingency(df2)
    df3 = pd.DataFrame(
            data=expected[:,:],
            index=df2.index,
            columns=df2.columns
    ).round(2)
    print('\nEffectif théorique par cellule :\n')
    print(df3)
    print(f'\np-value de : {pvalue:.5f}')
    if pvalue < 0.05:
        print(f' => Il y a un effet de la prise de Rituximab sur le facteur␣
 ↪{facteur}')
    else:
        print(f" => Pas d'effet du Rituximab sur le facteur {facteur}")
```

[51]:
```
for facteur in liste_Yes_No:
    teste_independance(facteur)
```

```
 - cas de RELAPSE :

Tableau de contingence :

RELAPSE  False  True
ritux
False      439    105
True       454    102

Effectif théorique par cellule :

RELAPSE   False   True
ritux
False    441.63  102.37
True     451.37  104.63

p-value de : 0.74253
 => Pas d'effet du Rituximab sur le facteur RELAPSE


 - cas de TOXICITY_GR34 :

Tableau de contingence :

TOXICITY_GR34  False  True
ritux
False            387    157
True             247    309

Effectif théorique par cellule :

TOXICITY_GR34   False   True
ritux
False          313.54  230.46
True           320.46  235.54

p-value de : 0.00000
 => Il y a un effet de la prise de Rituximab sur le facteur TOXICITY_GR34


 - cas de ATCD :
```

```
Tableau de contingence :

ATCD   False  True
ritux
False    348    196
True     141    415

Effectif théorique par cellule :

ATCD    False   True
ritux
False  241.83  302.17
True   247.17  308.83

p-value de : 0.00000
 => Il y a un effet de la prise de Rituximab sur le facteur ATCD
```

[52]:
```
for facteur in liste_oui_non[2:]:
    teste_independance(facteur)
```

```
 - cas de cv_totR2 :

Tableau de contingence :

cv_totR2  False  True
ritux
False       381    163
True        435    121

Effectif théorique par cellule :

cv_totR2   False   True
ritux
False     403.55  140.45
True      412.45  143.55

p-value de : 0.00238
 => Il y a un effet de la prise de Rituximab sur le facteur cv_totR2


 - cas de cv_valvR2 :

Tableau de contingence :

cv_valvR2  False  True
ritux
```

```
False        509     35
True         546     10

Effectif théorique par cellule :

cv_valvR2   False  True
ritux
False      521.75  22.25
True       533.25  22.75

p-value de : 0.00019
 => Il y a un effet de la prise de Rituximab sur le facteur cv_valvR2


 - cas de cv_rythR2 :

Tableau de contingence :

cv_rythR2  False  True
ritux
False        461     83
True         500     56

Effectif théorique par cellule :

cv_rythR2   False  True
ritux
False      475.26  68.74
True       485.74  70.26

p-value de : 0.01252
 => Il y a un effet de la prise de Rituximab sur le facteur cv_rythR2


 - cas de cv_insufcardR2 :

Tableau de contingence :

cv_insufcardR2  False  True
ritux
False             500     44
True              533     23

Effectif théorique par cellule :

cv_insufcardR2   False  True
ritux
False           510.87  33.13
```

```
True            522.13  33.87

p-value de : 0.00896
 => Il y a un effet de la prise de Rituximab sur le facteur cv_insufcardR2


 - cas de cv_arteriopathR2 :

Tableau de contingence :

cv_arteriopathR2  False  True
ritux
False               527     17
True                544     12

Effectif théorique par cellule :

cv_arteriopathR2   False  True
ritux
False             529.66  14.34
True              541.34  14.66

p-value de : 0.41659
 => Pas d'effet du Rituximab sur le facteur cv_arteriopathR2


 - cas de cv_avcR2 :

Tableau de contingence :

cv_avcR2  False  True
ritux
False       524     20
True        547      9

Effectif théorique par cellule :

cv_avcR2   False  True
ritux
False     529.66  14.34
True      541.34  14.66

p-value de : 0.05219
 => Pas d'effet du Rituximab sur le facteur cv_avcR2


 - cas de cv_thromboR2 :
```

```
Tableau de contingence :

cv_thromboR2  False  True
ritux
False           509     35
True            521     35

Effectif théorique par cellule :

cv_thromboR2   False  True
ritux
False         509.38  34.62
True          520.62  35.38

p-value de : 1.00000
 => Pas d'effet du Rituximab sur le facteur cv_thromboR2


 - cas de inf_totR2 :

Tableau de contingence :

inf_totR2  False  True
ritux
False        466     78
True         481     75

Effectif théorique par cellule :

inf_totR2   False  True
ritux
False      468.33  75.67
True       478.67  77.33

p-value de : 0.74919
 => Pas d'effet du Rituximab sur le facteur inf_totR2


 - cas de inf_zonaR2 :

Tableau de contingence :

inf_zonaR2  False  True
ritux
False         475     69
True          490     66

Effectif théorique par cellule :
```

```
inf_zonaR2   False  True
ritux
False       477.24  66.76
True        487.76  68.24

p-value de : 0.74963
 => Pas d'effet du Rituximab sur le facteur inf_zonaR2


 - cas de inf_verrueR2 :

Tableau de contingence :

inf_verrueR2  False  True
ritux
False           533     11
True            549      7

Effectif théorique par cellule :

inf_verrueR2  False  True
ritux
False         535.1    8.9
True          546.9    9.1

p-value de : 0.44745
 => Pas d'effet du Rituximab sur le facteur inf_verrueR2


 - cas de inf_hepbR2 :

Tableau de contingence :

inf_hepbR2  False  True
ritux
False         542      2
True          555      1

Effectif théorique par cellule :

inf_hepbR2   False  True
ritux
False       542.52   1.48
True        554.48   1.52

p-value de : 0.98490
 => Pas d'effet du Rituximab sur le facteur inf_hepbR2
```

```
 - cas de inf_hepcR2 :

Tableau de contingence :

inf_hepcR2  False  True
ritux
False         542      2
True          555      1

Effectif théorique par cellule :

inf_hepcR2   False  True
ritux
False       542.52   1.48
True        554.48   1.52

p-value de : 0.98490
 => Pas d'effet du Rituximab sur le facteur inf_hepcR2


 - cas de inf_tuberR2 :

Tableau de contingence :

inf_tuberR2  False  True
ritux
False          543      1
True           554      2

Effectif théorique par cellule :

inf_tuberR2   False  True
ritux
False        542.52   1.48
True         554.48   1.52

p-value de : 1.00000
 => Pas d'effet du Rituximab sur le facteur inf_tuberR2


 - cas de musclos_totR2 :

Tableau de contingence :

musclos_totR2  False  True
ritux
```

```
False            456     88
True             477     79

Effectif théorique par cellule :

musclos_totR2   False  True
ritux
False          461.41  82.59
True           471.59  84.41

p-value de : 0.40920
 => Pas d'effet du Rituximab sur le facteur musclos_totR2


 - cas de musclos_rayR2 :

Tableau de contingence :

musclos_rayR2  False  True
ritux
False            519     25
True             524     32

Effectif théorique par cellule :

musclos_rayR2   False  True
ritux
False          515.81  28.19
True           527.19  28.81

p-value de : 0.46441
 => Pas d'effet du Rituximab sur le facteur musclos_rayR2


 - cas de musclos_necroR2 :

Tableau de contingence :

musclos_necroR2  False  True
ritux
False              532     12
True               546     10

Effectif théorique par cellule :

musclos_necroR2   False  True
ritux
False            533.12  10.88
```

```
True             544.88  11.12

p-value de : 0.78942
 => Pas d'effet du Rituximab sur le facteur musclos_necroR2


 - cas de musclos_fibrocouR2 :

Tableau de contingence :

musclos_fibrocouR2  False  True
ritux
False                 543      1
True                  554      2

Effectif théorique par cellule :

musclos_fibrocouR2   False  True
ritux
False               542.52   1.48
True                554.48   1.52

p-value de : 1.00000
 => Pas d'effet du Rituximab sur le facteur musclos_fibrocouR2


 - cas de musclos_fibroautrR2 :

Tableau de contingence :

musclos_fibroautrR2  False  True
ritux
False                 538      6
True                  549      7

Effectif théorique par cellule :

musclos_fibroautrR2   False  True
ritux
False                537.57   6.43
True                 549.43   6.57

p-value de : 1.00000
 => Pas d'effet du Rituximab sur le facteur musclos_fibroautrR2


 - cas de musclos_arthroR2 :
```

```
Tableau de contingence :

musclos_arthroR2  False  True
ritux
False               484     60
True                509     47

Effectif théorique par cellule :

musclos_arthroR2   False  True
ritux
False             491.08  52.92
True              501.92  54.08

p-value de : 0.18030
 => Pas d'effet du Rituximab sur le facteur musclos_arthroR2


 - cas de poum_totR2 :

Tableau de contingence :

poum_totR2  False  True
ritux
False         476     68
True          488     68

Effectif théorique par cellule :

poum_totR2   False  True
ritux
False       476.74  67.26
True        487.26  68.74

p-value de : 0.96466
 => Pas d'effet du Rituximab sur le facteur poum_totR2


 - cas de poum_emboR2 :

Tableau de contingence :

poum_emboR2  False  True
ritux
False          533     11
True           544     12

Effectif théorique par cellule :
```

```
poum_emboR2   False  True
ritux
False        532.63  11.37
True         544.37  11.63

p-value de : 1.00000
 => Pas d'effet du Rituximab sur le facteur poum_emboR2


 - cas de poum_pleurR2 :

Tableau de contingence :

poum_pleurR2  False  True
ritux
False           529     15
True            541     15

Effectif théorique par cellule :

poum_pleurR2   False  True
ritux
False         529.16  14.84
True          540.84  15.16

p-value de : 1.00000
 => Pas d'effet du Rituximab sur le facteur poum_pleurR2


 - cas de poum_foncpulmR2 :

Tableau de contingence :

poum_foncpulmR2  False  True
ritux
False              522     22
True               535     21

Effectif théorique par cellule :

poum_foncpulmR2   False  True
ritux
False            522.73  21.27
True             534.27  21.73

p-value de : 0.94182
 => Pas d'effet du Rituximab sur le facteur poum_foncpulmR2
```

```
 - cas de poum_bpocR2 :

Tableau de contingence :

poum_bpocR2  False  True
ritux
False          527     17
True           542     14

Effectif théorique par cellule :

poum_bpocR2   False  True
ritux
False        528.67  15.33
True         540.33  15.67

p-value de : 0.67009
 => Pas d'effet du Rituximab sur le facteur poum_bpocR2


 - cas de poum_pneumoR2 :

Tableau de contingence :

poum_pneumoR2  False  True
ritux
False            523     21
True             536     20

Effectif théorique par cellule :

poum_pneumoR2   False  True
ritux
False          523.72  20.28
True           535.28  20.72

p-value de : 0.94324
 => Pas d'effet du Rituximab sur le facteur poum_pneumoR2


 - cas de k_totR2 :

Tableau de contingence :

k_totR2  False  True
ritux
```

```
False      473     71
True       513     43

Effectif théorique par cellule :

k_totR2   False  True
ritux
False    487.62  56.38
True     498.38  57.62

p-value de : 0.00520
 => Il y a un effet de la prise de Rituximab sur le facteur k_totR2


 - cas de autr_diabR2 :

Tableau de contingence :

autr_diabR2  False  True
ritux
False          495     49
True           522     34

Effectif théorique par cellule :

autr_diabR2   False  True
ritux
False        502.95  41.05
True         514.05  41.95

p-value de : 0.08882
 => Pas d'effet du Rituximab sur le facteur autr_diabR2


 - cas de bouche_totR2 :

Tableau de contingence :

bouche_totR2  False  True
ritux
False           376    168
True            413    143

Effectif théorique par cellule :

bouche_totR2  False  True
ritux
False         390.2  153.8
```

```
True          398.8  157.2

p-value de : 0.06663
 => Pas d'effet du Rituximab sur le facteur bouche_totR2


 - cas de bouche_prothR2 :

Tableau de contingence :

bouche_prothR2  False  True
ritux
False             438    106
True              473     83

Effectif théorique par cellule :

bouche_prothR2   False  True
ritux
False           450.53  93.47
True            460.47  95.53

p-value de : 0.05443
 => Pas d'effet du Rituximab sur le facteur bouche_prothR2


 - cas de bouche_goutR2 :

Tableau de contingence :

bouche_goutR2  False  True
ritux
False            487     57
True             502     54

Effectif théorique par cellule :

bouche_goutR2   False  True
ritux
False          489.11  54.89
True           499.89  56.11

p-value de : 0.74788
 => Pas d'effet du Rituximab sur le facteur bouche_goutR2


 - cas de bouche_secR2 :
```

```
Tableau de contingence :

bouche_secR2  False  True
ritux
False           482     62
True            517     39

Effectif théorique par cellule :

bouche_secR2   False  True
ritux
False         494.05  49.95
True          504.95  51.05

p-value de : 0.01585
 => Il y a un effet de la prise de Rituximab sur le facteur bouche_secR2


 - cas de bouche_compliR2 :

Tableau de contingence :

bouche_compliR2  False  True
ritux
False              533     11
True               553      3

Effectif théorique par cellule :

bouche_compliR2   False  True
ritux
False            537.08   6.92
True             548.92   7.08

p-value de : 0.05435
 => Pas d'effet du Rituximab sur le facteur bouche_compliR2


 - cas de dig_totR2 :

Tableau de contingence :

dig_totR2  False  True
ritux
False        513     31
True         523     33

Effectif théorique par cellule :
```

```
dig_totR2   False  True
ritux
False      512.35  31.65
True       523.65  32.35

p-value de : 0.96899
 => Pas d'effet du Rituximab sur le facteur dig_totR2


 - cas de dig_oesophR2 :

Tableau de contingence :

dig_oesophR2  False  True
ritux
False           530     14
True            546     10

Effectif théorique par cellule :

dig_oesophR2   False  True
ritux
False         532.13  11.87
True          543.87  12.13

p-value de : 0.50079
 => Pas d'effet du Rituximab sur le facteur dig_oesophR2


 - cas de dig_occluR2 :

Tableau de contingence :

dig_occluR2  False  True
ritux
False          535      9
True           547      9

Effectif théorique par cellule :

dig_occluR2  False  True
ritux
False        535.1    8.9
True         546.9    9.1

p-value de : 1.00000
 => Pas d'effet du Rituximab sur le facteur dig_occluR2
```

```
 - cas de dig_ulcR2 :

Tableau de contingence :

dig_ulcR2  False  True
ritux
False        529     15
True         542     14

Effectif théorique par cellule :

dig_ulcR2   False  True
ritux
False      529.66  14.34
True       541.34  14.66

p-value de : 0.95252
 => Pas d'effet du Rituximab sur le facteur dig_ulcR2


 - cas de dig_perfoR2 :

Tableau de contingence :

dig_perfoR2  False  True
ritux
False          540      4
True           555      1

Effectif théorique par cellule :

dig_perfoR2   False  True
ritux
False        541.53   2.47
True         553.47   2.53

p-value de : 0.35707
 => Pas d'effet du Rituximab sur le facteur dig_perfoR2


 - cas de autr_insufrenR2 :

Tableau de contingence :

autr_insufrenR2  False  True
ritux
```

```
False              510     34
True               532     24

Effectif théorique par cellule :

autr_insufrenR2   False  True
ritux
False            515.32  28.68
True             526.68  29.32

p-value de : 0.19372
 => Pas d'effet du Rituximab sur le facteur autr_insufrenR2


 - cas de autr_trsensR2 :

Tableau de contingence :

autr_trsensR2  False  True
ritux
False            439    105
True             444    112

Effectif théorique par cellule :

autr_trsensR2   False   True
ritux
False          436.68  107.32
True           446.32  109.68

p-value de : 0.78312
 => Pas d'effet du Rituximab sur le facteur autr_trsensR2


 - cas de autr_fatigR2 :

Tableau de contingence :

autr_fatigR2  False  True
ritux
False           420    124
True            418    138

Effectif théorique par cellule :

autr_fatigR2   False   True
ritux
False         414.43  129.57
```

```
True          423.57  132.43

p-value de : 0.47282
 => Pas d'effet du Rituximab sur le facteur autr_fatigR2


 - cas de autr_deprR2 :

Tableau de contingence :

autr_deprR2  False  True
ritux
False          476     68
True           507     49

Effectif théorique par cellule :

autr_deprR2   False  True
ritux
False        486.14  57.86
True         496.86  59.14

p-value de : 0.05939
 => Pas d'effet du Rituximab sur le facteur autr_deprR2


 - cas de autr_anxR2 :

Tableau de contingence :

autr_anxR2  False  True
ritux
False         453     91
True          478     78

Effectif théorique par cellule :

autr_anxR2   False  True
ritux
False       460.42  83.58
True        470.58  85.42

p-value de : 0.24703
 => Pas d'effet du Rituximab sur le facteur autr_anxR2


 - cas de autr_suicR2 :
```

```
Tableau de contingence :

autr_suicR2  False  True
ritux
False          533     11
True           548      8

Effectif théorique par cellule :

autr_suicR2  False  True
ritux
False        534.6    9.4
True         546.4    9.6

p-value de : 0.60946
 => Pas d'effet du Rituximab sur le facteur autr_suicR2
```

En conclusion, les effets de la prise de Rituximab se font ressentir sur cv_totR2, cv_valvR2, cv_rythR2, cv_insufcardR2, k_totR2, et bouche_secR2, avec à chaque fois moins de cas qu'attendu (effet protecteur du Rituximab ?). Par exemple, dans le cas de l'insuffisance cardiaque, le tableau de contingence est :

[53]:
```
facteur = 'cv_insufcardR2'
df2 = pd.crosstab(df['ritux'],df[facteur])
print(df2)
```

```
cv_insufcardR2  False  True
ritux
False             500     44
True              533     23
```

dans la ligne False des patients n'ayant pas été traités par le Rituximab, on voit qu'il y a bien plus d'insuffisance cardiaque (44) que dans celle des patients traités par Rituximab (23), quand les patients traités et non traités par ce médicament sont sensiblement les mêmes. Dans une situation d'indépendance, voici ce qui aurait été attendu :

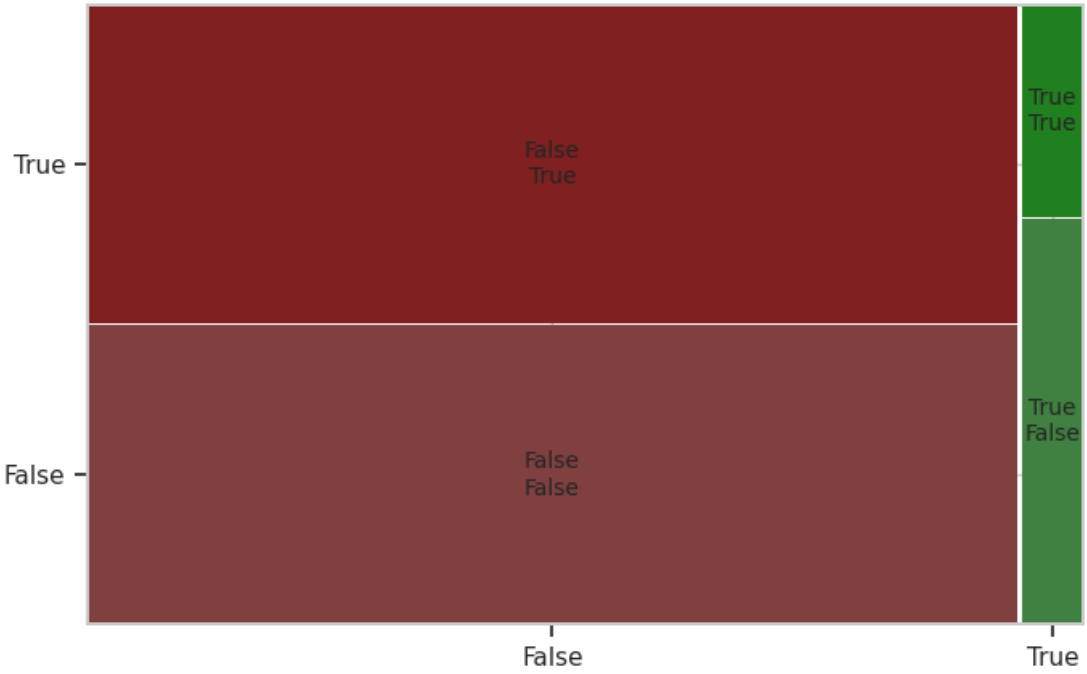
[54]:
```
X2value, pvalue, degree_of_freedom, expected = chi2_contingency(df2)
pd.DataFrame(
        data=expected[:,:],
        index=df2.index,
        columns=df2.columns
).round(2)
```

[54]:
```
cv_insufcardR2   False  True
ritux
False           510.87  33.13
True            522.13  33.87
```

On aurait dû avoir à peu près autant de cas d'insuffisance cardiaque (33) dans les deux cas, et non

cette forte disymétrie, visualisable dans le diagramme en mosaïque :

[55]:
```
mosaic(df, [facteur, 'ritux'], axes_label=True);
```




Enfin, la p-value est de :

[56]:
```
pvalue
```

[56]:
```
0.008957200846988823
```

Elle est inférieure à 0.05, et même à 0.01.

Il semblerait donc qu'il y ait un effet Rituximab bénéfique, principalement sur les maladies cardio-vasculaires. Mais cet effet ne se ressent pas au niveau du coût total. Quelques explications possibles : - Il s'agit-là d'effets à la marge, pas suffisants pour impacter le coût total des traitements après guérison du lymphome. - Il y a moins de maladies cardio-vasculaires, mais elles sont plus graves, plus coûteuses. - Le coût total est imprécis (?)

[57]:
```
cas = 'cv_insufcardR2'
print(df.loc[df.ritux==False].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.
  ↪ritux==False].loc[df[cas]].COUT_TOTAL.mean())
print(df.loc[df.ritux].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.ritux].
  ↪loc[df[cas]].COUT_TOTAL.mean())
```

```
5.933951869970813 6.76020830323007
5.818640986136308 7.054423351827457
```

[58]:
```
cas = 'cv_totR2'
print(df.loc[df.ritux==False].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.
  ↪ritux==False].loc[df[cas]].COUT_TOTAL.mean())
print(df.loc[df.ritux].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.ritux].
  ↪loc[df[cas]].COUT_TOTAL.mean())
```

```
5.800488351502913 6.468951155858408
5.733962668042329 6.357963819043559
```

[59]:
```
cas = 'cv_valvR2'
print(df.loc[df.ritux==False].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.
  ↪ritux==False].loc[df[cas]].COUT_TOTAL.mean())
print(df.loc[df.ritux].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.ritux].
  ↪loc[df[cas]].COUT_TOTAL.mean())
```

```
5.954274109440162 6.677130817785368
5.852836723677806 6.793853157460184
```

[60]:
```
cas = 'cv_rythR2'
print(df.loc[df.ritux==False].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.
  ↪ritux==False].loc[df[cas]].COUT_TOTAL.mean())
print(df.loc[df.ritux].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.ritux].
  ↪loc[df[cas]].COUT_TOTAL.mean())
```

```
5.879402647508337 6.6749455400745346
5.8341411567531685 6.187800077251766
```

[61]:
```
cas = 'k_totR2'
print(df.loc[df.ritux==False].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.
  ↪ritux==False].loc[df[cas]].COUT_TOTAL.mean())
print(df.loc[df.ritux].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.ritux].
  ↪loc[df[cas]].COUT_TOTAL.mean())
```

```
5.945466194160702 6.369290006894611
5.820553444168688 6.456824787073182
```

[62]:
```
cas = 'bouche_secR2'
print(df.loc[df.ritux==False].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.
  ↪ritux==False].loc[df[cas]].COUT_TOTAL.mean())
print(df.loc[df.ritux].loc[df[cas]==False].COUT_TOTAL.mean(), df.loc[df.ritux].
  ↪loc[df[cas]].COUT_TOTAL.mean())
```

```
5.94178747368548 6.459411903405302
5.840758630906296 6.254235141644317
```

## 1.5 Approche machine learning

On regarde dans quelle mesure un modèle récent et performant de régression à base de collections d'arbres de décision permet d'apprendre à retrouver le coût total à partir de l'ensemble des variables explicatives. Et l'on regardera ensuite l'importance de chaque variable dans cette prédiction, histoire de voir si la présence ou non de modèle dans le Rituximab a été utile à la prédiction.

[63]:
```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(df, test_size = 0.2, random_state = 42)
```

[64]:
```
target = 'COUT_TOTAL'
#target = 'ritux'

X_train = train_set.drop(target, axis = 1)
y_train = train_set[target].copy()

X_test = test_set.drop(target, axis = 1)
y_test = test_set[target].copy()
```

### 1.5.1 Arbre de décision

[65]:
```
from sklearn.tree import DecisionTreeRegressor
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

[66]:
```
for k in range(1,10):
    reg = DecisionTreeRegressor(max_depth=k)
    reg.fit(X_train, y_train)
    y_test_pred = reg.predict(X_test)
    test_error = mean_squared_error(y_test, y_test_pred)
    print(f"Max depth: {k}")
    print("  - Validation RMSE:", np.sqrt(test_error))
    print("  - Validation MAE:", mean_absolute_error(y_test_pred, y_test))
```

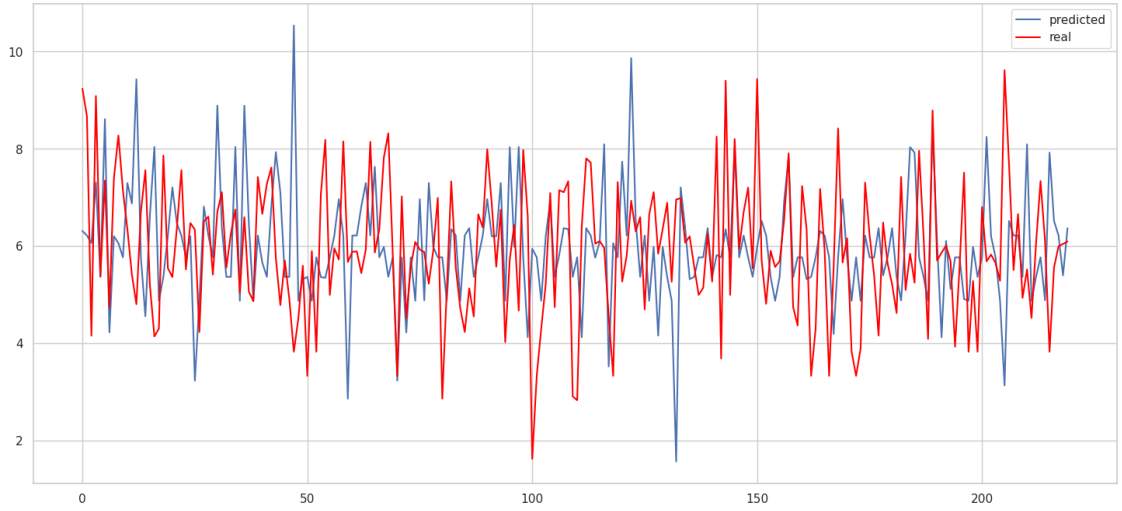
```
Max depth: 1
  - Validation RMSE: 1.3640461473410157
  - Validation MAE: 1.0666122346855236
Max depth: 2
  - Validation RMSE: 1.3372576372383116
  - Validation MAE: 1.064989619436803
Max depth: 3
  - Validation RMSE: 1.402649641168311
  - Validation MAE: 1.0942874602401707
Max depth: 4
  - Validation RMSE: 1.4200089302959795
  - Validation MAE: 1.0946777370763867
Max depth: 5
  - Validation RMSE: 1.5018274352296028
```

```
  - Validation MAE: 1.15172691036932
Max depth: 6
  - Validation RMSE: 1.5609177840748354
  - Validation MAE: 1.1885814091796527
Max depth: 7
  - Validation RMSE: 1.6408332692782608
  - Validation MAE: 1.2659940908655414
Max depth: 8
  - Validation RMSE: 1.661110630899666
  - Validation MAE: 1.2512095111092822
Max depth: 9
  - Validation RMSE: 1.7026883337686642
  - Validation MAE: 1.2953250969260728
```

[67]:
```
%pylab inline
pylab.rcParams['figure.figsize'] = (18,8)

N = len(y_test_pred)
plt.plot(range(N), y_test_pred, label="predicted")
plt.plot(range(N), y_test, color='red', label="real")
plt.legend()
```

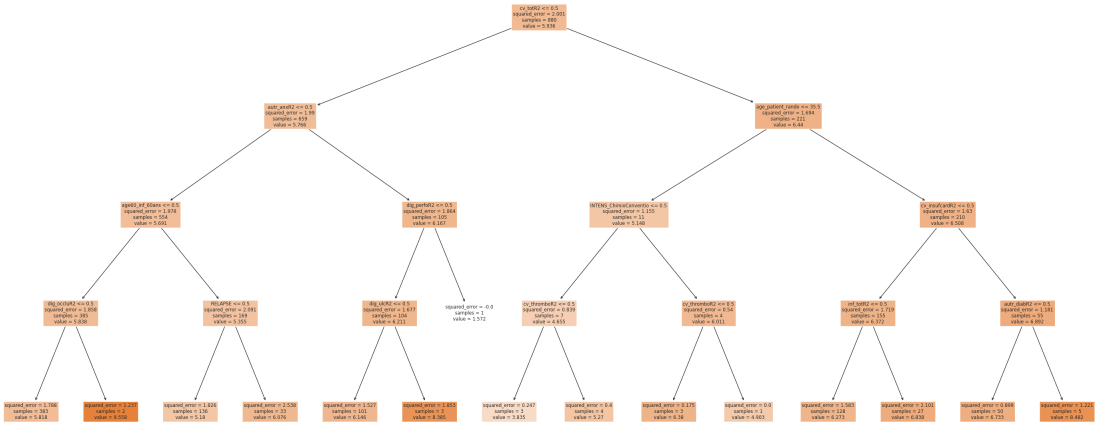
```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```

[67]:
```
<matplotlib.legend.Legend at 0x7f04c9598520>
```




[68]:
```
from sklearn import tree
pylab.rcParams['figure.figsize'] = (36,16)
```

```
reg = DecisionTreeRegressor(max_depth=4)
reg.fit(X_train, y_train)

tree.plot_tree(reg, filled=True, feature_names=list(df.columns))
plt.savefig('decision_tree.pdf')
```



### 1.6 Version XGBoost

[69]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
for col in liste_nb:
    df[col] = scaler.fit_transform(df[col].values.reshape(-1, 1)).flatten()
```

[70]:
```
import xgboost
target = 'cv_insufcardR2'
#target = 'ritux'

df = df.drop('COUT_TOTAL', axis=1)
train_set, test_set = train_test_split(df, test_size = 0.2, random_state = 42)

X_train = train_set.drop(target, axis = 1)
y_train = train_set[target].copy()

X_test = test_set.drop(target, axis = 1)
y_test = test_set[target].copy()

xgb_reg = xgboost.XGBRegressor(learning_rate = 0.01,
                               max_depth = 6,
                               random_state=42,
                               n_estimators = 600,
```

```
                                        n_jobs=-1
                                    )

xgb_reg.fit(X_train, y_train)

y_test_pred = xgb_reg.predict(X_test)
print(sqrt(mean_squared_error(y_test_pred, y_test)),␣
 ↪mean_absolute_error(y_test_pred,y_test))
```

```
0.20287934 0.07751636
```

[71]:
```
%pylab inline
pylab.rcParams['figure.figsize'] = (10, 30)
xgboost.plot_importance(xgb_reg)
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```

[71]:
```
<AxesSubplot: title={'center': 'Feature importance'}, xlabel='F score',
ylabel='Features'>
```


Feature importance
age_patient_rando 3275.0
cv_totR2 980.0
cv_rythR2 803.0
FOLLOW_UP 753.0
cv_arteriopathR2 635.0
cv_thromboR2 555.0
cv_valvR2 485.0
cv_avcR2 447.0
autr_fatigR2 332.0
RELAPSE_TRT_Chemotherapy,Immunotherapy 323.0
greffe 323.0
bouche_totR2 229.0
autr_insufrenR2 223.0
poum_pneumoR2 204.0
musclos_arthroR2 180.0
bouche_prothR2 176.0
k_totR2 170.0
autr_trsensR2 166.0
autr_diabR2 166.0
INTENS_ChimioConventio 155.0
inf_verrueR2 155.0
sexeR_FEMALE 152.0
bouche_secR2 141.0
INTENS_ChimioIntensive 140.0
INTENS_ChimioIntensive+Greffe 135.0
poum_totR2 133.0
ATCD 125.0
dig_totR2 114.0
bouche_goutR2 108.0
autr_anxR2 106.0
poum_emboR2 102.0
RELAPSE 100.0
TOXICITY_GR34 98.0
musclos_totR2 77.0
inf_totR2 76.0
poum_foncpulmR2 70.0
musclos_necroR2 70.0
inf_hepbR2 68.0
dig_ulcR2 61.0
bouche_compliR2 57.0
musclos_fibroautrR2 54.0
ritux 54.0
RELAPSE_TRT_Radiotherapy,OtherTreatment 45.0
RELAPSE_TRT_Chemotherapy 42.0
autr_suicR2 42.0
dig_oesophR2 41.0
autr_deprR2 37.0
age60_inf_60ans 35.0
inf_zonaR2 28.0
musclos_rayR2 20.0
dig_occluR2 1.0
Features
0
500
1000
1500
2000
2500
3000
3500
F score

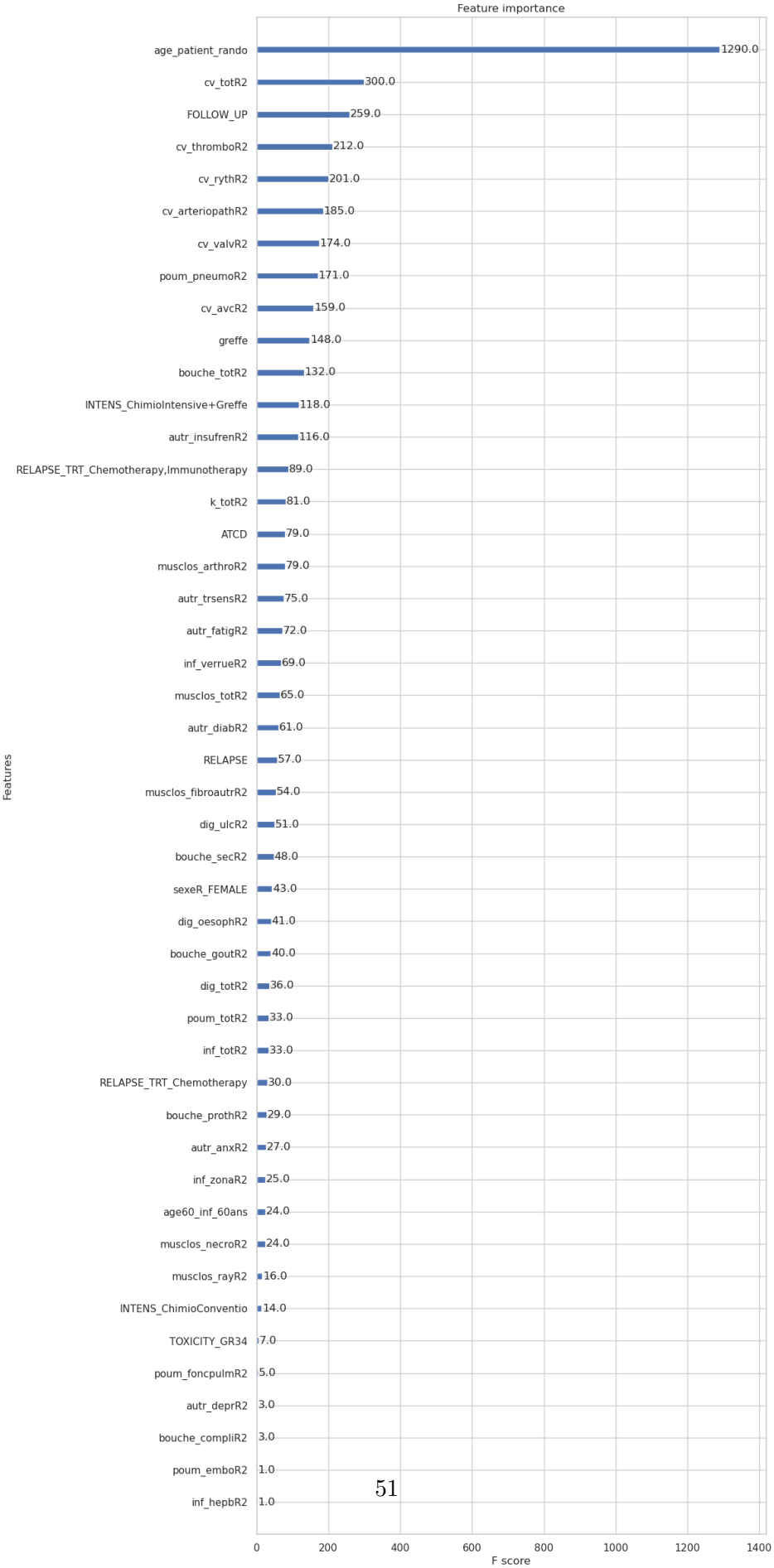
[72]:
```
target = 'cv_insufcardR2'

df1 = df.drop('ritux', axis=1)
train_set, test_set = train_test_split(df1, test_size = 0.2, random_state = 42)

X_train = train_set.drop(target, axis = 1)
y_train = train_set[target].copy()

X_test = test_set.drop(target, axis = 1)
y_test = test_set[target].copy()

xgb_reg = xgboost.XGBRegressor(learning_rate = 0.01,
                                        max_depth = 6,
                                        random_state=42,
                                        n_estimators = 300,
                                        n_jobs=-1
                                )

xgb_reg.fit(X_train, y_train)

y_test_pred = xgb_reg.predict(X_test)
print(sqrt(mean_squared_error(y_test_pred, y_test)),␣
 ↪mean_absolute_error(y_test_pred,y_test))
```

```
0.20863071 0.10498003
```

[73]:
```
%pylab inline
pylab.rcParams['figure.figsize'] = (10, 30)
xgboost.plot_importance(xgb_reg)
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```

[73]:
```
<AxesSubplot: title={'center': 'Feature importance'}, xlabel='F score',
ylabel='Features'>
```


Feature importance
age_patient_rando 1290.0
cv_totR2 300.0
FOLLOW_UP 259.0
cv_thromboR2 212.0
cv_rythR2 201.0
cv_arteriopathR2 185.0
cv_valvR2 174.0
poum_pneumoR2 171.0
cv_avcR2 159.0
greffe 148.0
bouche_totR2 132.0
INTENS_ChimioIntensive+Greffe 118.0
autr_insufrenR2 116.0
RELAPSE_TRT_Chemotherapy,Immunotherapy 89.0
k_totR2 81.0
ATCD 79.0
musclos_arthroR2 79.0
autr_trsensR2 75.0
autr_fatigR2 72.0
inf_verrueR2 69.0
musclos_totR2 65.0
autr_diabR2 61.0
RELAPSE 57.0
musclos_fibroautrR2 54.0
dig_ulcR2 51.0
bouche_secR2 48.0
sexeR_FEMALE 43.0
dig_oesophR2 41.0
bouche_goutR2 40.0
dig_totR2 36.0
poum_totR2 33.0
inf_totR2 33.0
RELAPSE_TRT_Chemotherapy 30.0
bouche_prothR2 29.0
autr_anxR2 27.0
inf_zonaR2 25.0
age60_inf_60ans 24.0
musclos_necroR2 24.0
musclos_rayR2 16.0
INTENS_ChimioConventio 14.0
TOXICITY_GR34 7.0
poum_foncpulmR2 5.0
autr_deprR2 3.0
bouche_compliR2 3.0
poum_emboR2 1.0
inf_hepbR2 1.0
Features
0
200
400
600
800
1000
1200
1400
F score

[74]:
```
target = 'cv_totR2'

train_set, test_set = train_test_split(df, test_size = 0.2, random_state = 42)

X_train = train_set.drop(target, axis = 1)
y_train = train_set[target].copy()

X_test = test_set.drop(target, axis = 1)
y_test = test_set[target].copy()

xgb_reg = xgboost.XGBRegressor(learning_rate = 0.01,
                                        max_depth = 6,
                                        random_state=42,
                                        n_estimators = 600,
                                        n_jobs=-1
                                )

xgb_reg.fit(X_train, y_train)

y_test_pred = xgb_reg.predict(X_test)
print(sqrt(mean_squared_error(y_test_pred, y_test)),␣
 ↪mean_absolute_error(y_test_pred,y_test))
```

```
0.0012586148 0.0012546075
```

[75]:
```
df1 = df.drop('ritux', axis=1)
train_set, test_set = train_test_split(df1, test_size = 0.2, random_state = 42)

X_train = train_set.drop(target, axis = 1)
y_train = train_set[target].copy()

X_test = test_set.drop(target, axis = 1)
y_test = test_set[target].copy()

xgb_reg = xgboost.XGBRegressor(learning_rate = 0.01,
                                        max_depth = 6,
                                        random_state=42,
                                        n_estimators = 300,
                                        n_jobs=-1
                                )

xgb_reg.fit(X_train, y_train)

y_test_pred = xgb_reg.predict(X_test)
```

```
print(sqrt(mean_squared_error(y_test_pred, y_test)),␣
 ↪mean_absolute_error(y_test_pred,y_test))
```

```
0.025036488 0.025020266
```

[76]:
```
# On supprime tous les autres cv_
target = 'cv_insufcardR2'

df1 = df.copy()

for k in df.columns:
    if k.startswith('cv_') and k != target:
        df1 = df1.drop(k, axis=1)
#df1 = df.drop('ritux', axis=1)

train_set, test_set = train_test_split(df1, test_size = 0.2, random_state = 42)

X_train = train_set.drop(target, axis = 1)
y_train = train_set[target].copy()

X_test = test_set.drop(target, axis = 1)
y_test = test_set[target].copy()

xgb_reg = xgboost.XGBRegressor(learning_rate = 0.01,
                                       max_depth = 6,
                                       random_state=42,
                                       n_estimators = 300,
                                       n_jobs=-1
                                )

xgb_reg.fit(X_train, y_train)

y_test_pred = xgb_reg.predict(X_test)
print(sqrt(mean_squared_error(y_test_pred, y_test)),␣
 ↪mean_absolute_error(y_test_pred,y_test))
```

```
0.25327423 0.13119614
```

[77]:
```
# On supprime tous les autres cv_
target = 'cv_insufcardR2'

df1 = df.copy()

for k in df.columns:
    if k.startswith('cv_') and k != target:
        df1 = df1.drop(k, axis=1)
df1 = df1.drop('ritux', axis=1)
```

```
train_set, test_set = train_test_split(df1, test_size = 0.2, random_state = 42)

X_train = train_set.drop(target, axis = 1)
y_train = train_set[target].copy()

X_test = test_set.drop(target, axis = 1)
y_test = test_set[target].copy()

xgb_reg = xgboost.XGBRegressor(learning_rate = 0.01,
                                       max_depth = 6,
                                       random_state=42,
                                       n_estimators = 300,
                                       n_jobs=-1
                               )

xgb_reg.fit(X_train, y_train)

y_test_pred = xgb_reg.predict(X_test)
print(sqrt(mean_squared_error(y_test_pred, y_test)),␣
 ↪mean_absolute_error(y_test_pred,y_test))
```

```
0.25078818 0.13073494
```