1 Introduction

Pratique de la Régression ZIP (Zero-Inflated Poisson Regression) sous R et Python avec l'utilisation, respectivement, des packages ''pscl'' et ''statsmodels''.

Ce tutoriel fait suite au support de cours consacré à la ''Zero-Inflated Poisson Regression'' [**COURS ZIP,** Régression ZIP], une technique adaptée à la modélisation d'une variable de comptage lorsque la valeur ''0'' est surreprésentée.

Nous travaillerons sous R dans un premier temps. Nous détaillons les différentes manières de modéliser une variable cible représentant un dénombrement. Nous appliquerons tour à tour la régression logistique, la régression de Poisson et la régression ZIP avec le package ''pscl'' (Political Science Computational Library). Nous analyserons les résultats pour essayer de comprendre l'intérêt des différentes approches. Dans un deuxième temps, nous reprenons dans les grandes lignes la même étude en travaillant sous Python cette fois-ci. Nous ferons appel au package ''statsmodels''. Nous constaterons – sans surprise – la convergence des résultats avec ceux de R.

2 Données

Nous utilisons une variante des données ''Extramarital Affairs Data'' (Fair, 1977). L'objectif est d'expliquer le nombre de tromperies commises par une personne durant l'année écoulée. La variable dépendante ''Affairs (extra-maritales)'' (relations amoureuses, dixit le dictionnaire) du fichier ''**infidelite_zip_reg.xlslx**'' prend des valeurs entières comprises entre 0 et 12 (il s'agit plutôt d'un codage spécifique du dénombrement, mais nous considérerons dans cette étude qu'il s'agit d'un simple comptage). Pour la modéliser, nous nous appuyons sur les caractéristiques des personnes recensées (n = 601 observations), à savoir :

Gender (0 = female, 1 = male)
Age in years (mean = 32.5)
Number of years married
Children (0 = no, 1 = yes)
Religiousness (1 = anti to 5 = very)
Education in years
Occupation ("Hollingshead Scale" 1-7)
Self rating of marriage (1 = very unhappy to 5 = very happy)

La difficulté de l'analyse réside dans l'ambivalence de l'absence d'adultère (Affairs == 0). Elle peut traduire la fidélité à toute épreuve de la personne, quelle que soit la tentation à laquelle elle a été soumise. Comme elle peut résulter, pour la personne volage, d'un défaut d'opportunité durant la

période étudiée (*ouh là là, ça a été dur de résister à la tentation d'écrire quelque chose d'égrillard sur le sujet, mais le terrain est glissant, j'ai préféré rester sobre… un petit peu quand-même ? non, non, vraiment*). La régression ZIP permet d'appréhender la double nature de la configuration ''Affairs == 0'' en combinant la régression logistique (pour distinguer la fidélité de la bagatelle) avec la régression de Poisson (pour comptabiliser le nombre de batifolages chez les adeptes de la gaudriole, y compris la valeur 0).

3 Traitements sous R avec ''pscl''

3.1 Importation et préparation des données

Nous importons le fichier ''**infidelite_zip_reg.xlsx**'' et nous en affichons les caractéristiques. Nous utilisons le package ''xlsx'' pour manipuler le format Excel.

```
#changer de répertoire
setwd("... votre dossier ...")

#importation des données – format de fichier Excel
library(xlsx)
D <- read.xlsx("infidelite_zip_reg.xlsx",sheetIndex=1)
print(str(D))
```

```
'data.frame':601 obs. of  9 variables:
 $ Affairs       : num  0 0 0 0 0 0 0 0 7 0 ...
 $ Gender        : num  0 1 0 1 0 0 1 1 0 0 ...
 $ Age           : num  32 32 22 57 27 52 27 37 32 27 ...
 $ YearsMarried  : num  15 10 0.75 15 4 15 4 15 15 1.5 ...
 $ Children      : num  1 1 0 1 1 1 1 0 1 0 ...
 $ Religiousness : num  4 4 1 2 1 3 3 4 3 2 ...
 $ Education     : num  14 20 14 20 16 16 20 20 14 16 ...
 $ Occupation    : num  1 6 4 6 5 5 6 6 3 6 ...
 $ RatingMarriage: num  5 4 5 4 5 4 5 5 2 4 ...
```

Nous disposons de 601 observations et 9 variables, dont l'endogène ''Affairs'' qui prend les valeurs 0, 1, 2, 3, 7 et 12.

```
#barplot
nbAffairs <- table(D$Affairs)
barplot(nbAffairs)
```

Nous constatons que la valeur ''0'' est surreprésentée dans le décompte des liaisons extra-maritales. Elle représente 451 observations, soit 75% du fichier.

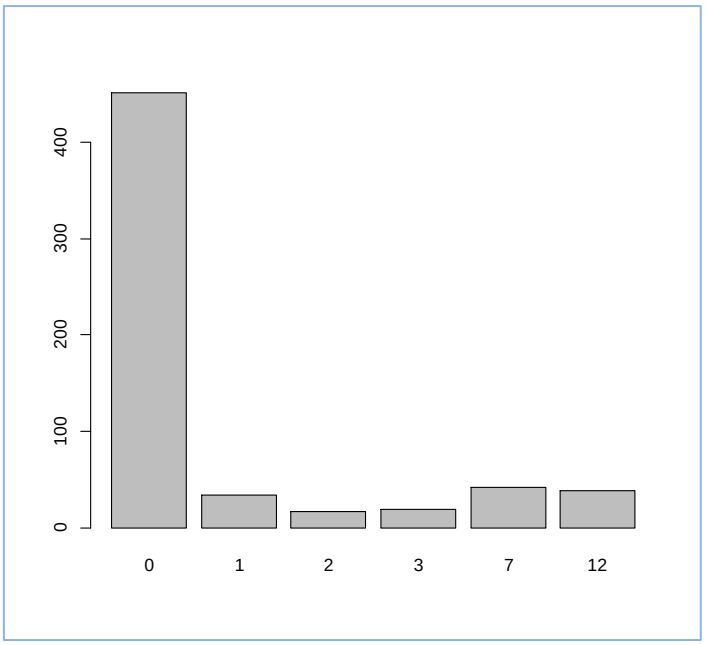


Figure 1 : Distribution de la variable dépendante ''Affairs''

Nous décidons de partitionner les données en échantillons d'apprentissage (301 individus) et de test (300). Ce n'est pas vraiment l'usage en économétrie. Je le fais parce que nous en aurons l'utilité pour comparer les performances des modèles avec le test de Vuong plus loin (section 3.5).

Parce que j'ai pris soin de mélanger aléatoirement les lignes dans le fichier Excel, nous prenons pour l'apprentissage (TRAIN) les 301 premières observations, et pour le test (TEST) les 300 suivantes. En effectuant la même partition sous Python (section 4), nous pourrons ainsi comparer directement les résultats.

```
#partition - sélection apprentissage – 301 premières
DTrain <- D[1:301,]

#sélection test – 300 suivantes
DTest <- D[302:601,]
```

3.2 Régression logistique

Dans une première analyse, nous essayons d'identifier les raisons qui poussent les personnes à folâtrer. Nous transformons la variable cible en absence (Affairs == 0), que nous codons ''1'', et présence de liaison (Affairs > 0), codé ''0''. J'ai fait le choix de ce codage inhabituel (l'absence du phénomène est représentée par ''1'') pour être en conformité avec celui adopté automatiquement par la régression ZIP (section 3.4).

Nous appliquons ensuite la régression logistique avec la commande `glm()` du package ''stats'' :

```
#codage binaire
YBIN <- ifelse(DTrain$Affairs == 0, 1, 0)

#régression logistique
mlr <- glm(YBIN ~ Gender+Age+YearsMarried+Children+Religiousness+Education+
Occupation+RatingMarriage, data = DTrain, family = binomial)
print(summary(mlr))
```

```
Call:
glm(formula = YBIN ~ Gender + Age + YearsMarried + Children +
    Religiousness + Education + Occupation + RatingMarriage,
    family = binomial, data = DTrain)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3731   0.3725   0.5565   0.6995   1.4899

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.20442    1.32543  -0.154  0.87743
Gender         -0.25520    0.36543  -0.698  0.48496
Age             0.04058    0.02844   1.427  0.15367
YearsMarried   -0.07727    0.04986  -1.550  0.12116
Children       -0.37014    0.42607  -0.869  0.38499
Religiousness   0.33261    0.13060   2.547  0.01087 *
Education      -0.06545    0.07679  -0.852  0.39398
Occupation     -0.05783    0.10680  -0.541  0.58818
RatingMarriage  0.41302    0.13309   3.103  0.00191 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 314.08  on 300  degrees of freedom
Residual deviance: 287.21  on 292  degrees of freedom
AIC: 305.21

Number of Fisher Scoring iterations: 4
```

''Religiousness'' (religiosité, 1 pas du tout, 5 cul-bénit) et ''RatingMarriage'' (satisfaction dans son mariage, 1 malheureux, 5 heureux) sont les deux variables qui semblent se démarquer au risque 5% (test de significativité individuelle, p-value = P(>|z|) est inférieur à 5%). Nous procédons à une sélection de variable avec `stepAIC()` pour en avoir le cœur net.

```
#sélection de variables - optimisation du critère BIC (k = log(n))
library(MASS)
mlrs <- stepAIC(mlr,direction="backward",k=log(nrow(DTrain)))
print(summary(mlrs))
```

```
Start:  AIC=338.57
YBIN ~ Gender + Age + YearsMarried + Children + Religiousness +
    Education + Occupation + RatingMarriage

                Df Deviance    AIC
- Occupation     1   287.50 333.16 #le retrait d'Occupation est celui qui réduit le plus le critère (338.57 -> 333.16)
```

```
- Gender          1   287.70 333.35
- Education       1   287.94 333.60
- Children        1   287.98 333.63
- Age             1   289.33 334.99
- YearsMarried    1   289.62 335.28
<none>                287.21 338.57
- Religiousness   1   293.86 339.52
- RatingMarriage  1   296.85 342.51

Step:  AIC=333.16
YBIN ~ Gender + Age + YearsMarried + Children + Religiousness +
    Education + RatingMarriage

                 Df Deviance    AIC
- Children        1   288.18 328.13
- Gender          1   288.35 328.30
- Education       1   288.85 328.80
- Age             1   289.55 329.50
- YearsMarried    1   289.90 329.85
<none>                287.50 333.16
- Religiousness   1   294.07 334.02
- RatingMarriage  1   297.41 337.36

Step:  AIC=328.13
YBIN ~ Gender + Age + YearsMarried + Religiousness + Education +
    RatingMarriage

                 Df Deviance    AIC
- Gender          1   289.11 323.35
- Education       1   289.53 323.77
- Age             1   290.53 324.78
- YearsMarried    1   292.48 326.73
<none>                288.18 328.13
- Religiousness   1   294.67 328.91
- RatingMarriage  1   298.56 332.80

Step:  AIC=323.35
YBIN ~ Age + YearsMarried + Religiousness + Education + RatingMarriage

                 Df Deviance    AIC
- Age             1   290.85 319.39
- Education       1   292.09 320.62
- YearsMarried    1   292.90 321.44
<none>                289.11 323.35
- Religiousness   1   295.67 324.21
- RatingMarriage  1   299.78 328.31

Step:  AIC=319.39
YBIN ~ YearsMarried + Religiousness + Education + RatingMarriage

                 Df Deviance    AIC
- YearsMarried    1   292.95 315.78
- Education       1   293.01 315.84
<none>                290.85 319.39
- Religiousness   1   297.87 320.70
- RatingMarriage  1   301.49 324.32

Step:  AIC=315.78
YBIN ~ Religiousness + Education + RatingMarriage
```

```
                 Df Deviance    AIC
- Education       1   295.15 312.27
- Religiousness   1   298.37 315.49
<none>                292.95 315.78
- RatingMarriage  1   307.30 324.42

Step:  AIC=312.27
YBIN ~ Religiousness + RatingMarriage

                 Df Deviance    AIC
- Religiousness   1   300.81 312.22 # Religiousness est éliminée d'un souffle
<none>                295.15 312.27
- RatingMarriage  1   308.29 319.71

Step:  AIC=312.22
YBIN ~ RatingMarriage

                 Df Deviance    AIC
<none>                300.81 312.22
- RatingMarriage  1   314.08 319.79
```

Aucun retrait de variable ne permet d'améliorer (réduire) le critère. Le processus de sélection est stoppé.

```
> print(summary(mlrs))

Call:
glm(formula = YBIN ~ RatingMarriage, family = binomial, data = DTrain)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9818   0.5499   0.5499   0.6727   1.1585

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.3973     0.4693  -0.847 0.397206
RatingMarriage   0.4420     0.1214   3.641 0.000272 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 314.08  on 300  degrees of freedom
Residual deviance: 300.81  on 299  degrees of freedom
AIC: 304.81

Number of Fisher Scoring iterations: 4
```

Avec l'option (direction = ''backward''), la variable ''Occuaption'' a été la première retirée, puis ''children'', etc. Seule ''RatingMarriage'' a été conservée finalement. Nous remarquerons cependant que ''Religiousness'' a été éliminée d'un souffle, le critère BIC (désigné par AIC dans l'affichage de R) étant passé de 312.27 à 312.22.

La seule variable pertinente pour expliquer le passage à l'acte serait donc ''RatingMarriage'' : plus nous sommes heureux en ménage, moins nous sommes enclins à voir ailleurs (YBIN == 1 représente les personnes qui n'ont pas fauté). La conclusion paraît sensée.

3.3 Régression de Poisson

Nous traitons ''Affairs'' directement comme une variable de comptage ici. Nous faisons de nouveau appel à `glm()` mais avec l'option (family = poisson).

```
#régression de Poisson
mpr <- glm(Affairs ~ ., data = DTrain, family = poisson)
print(summary(mpr))
```

```
Call:
glm(formula = Affairs ~ ., family = poisson, data = DTrain)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-3.889  -1.421  -1.086  -0.760   6.796

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     2.99251    0.45668   6.553 5.65e-11 ***
Gender          0.15246    0.13525   1.127    0.260
Age            -0.04905    0.01102  -4.451 8.56e-06 ***
YearsMarried    0.09883    0.01832   5.396 6.81e-08 ***
Children        0.04881    0.16683   0.293    0.770
Religiousness  -0.34758    0.04842  -7.179 7.03e-13 ***
Education       0.01962    0.02770   0.708    0.479
Occupation      0.04208    0.03751   1.122    0.262
RatingMarriage -0.47535    0.04527 -10.500  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1317.9  on 300  degrees of freedom
Residual deviance: 1064.9  on 292  degrees of freedom
AIC: 1293.4
```

Avec la sélection de variables ….

```
#sélection de variables – critère BIC – k = log(n_train)
mprs <- stepAIC(mpr,direction="backward",k=log(nrow(DTrain)))
print(summary(mprs))
```

```
Call:
glm(formula = Affairs ~ Age + YearsMarried + Religiousness +
    Occupation + RatingMarriage, family = poisson, data = DTrain)

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     3.12481    0.30338  10.300  < 2e-16 ***
Age            -0.04412    0.01035  -4.264 2.00e-05 ***
YearsMarried    0.09555    0.01615   5.915 3.31e-09 ***
Religiousness  -0.35175    0.04840  -7.268 3.65e-13 ***
Occupation      0.07484    0.03060   2.446   0.0144 *
RatingMarriage -0.46420    0.04338 -10.701  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1317.9  on 300  degrees of freedom
Residual deviance: 1067.5  on 295  degrees of freedom
AIC: 1290
```

… plusieurs explicatives se démarquent à 5% : ''RatingMarriage'', comme pour la régression logistique (le coefficient est négatif, avec cette nouvelle définition de la variable cible, plus on est heureux, moins on cherche ailleurs), mais aussi ''Age'' (influence négative sur l'adultère), ''YearsMarried'' (positive), ''Religiousness'' (négative) et ''Occupation'' (positive). Je ne me risquerai surtout pas à des interprétations hasardeuses.

Pour en revenir à la technique, on note une forte surdispersion avec le ratio entre ''Residual Deviance'' (1067.5) et ''degrees of freedom'' (295) qui est largement supérieur à 1 (3.62 pour être précis), conduisant à une sous-estimation des écarts-type des coefficients (**COURS POISSON**, page 30). Le résultat est faussé. La liste des variables réellement pertinentes (avec des coefficients significativement différents de 0 au risque 5%) est réduite en réalité. Passons à la Régression ZIP pour dépasser cet écueil.

3.4 Régression ZIP

Nous utilisons le package ''pscl'' pour la régression ZIP. Il y a une double modélisation : une première régression logistique binaire pour (''Affairs'' == 0) contre les autres (valeurs) avec un ensemble de variables Z, une seconde régression de Poisson (comptage de ''Affairs'') avec un ensemble de variables X (**COURS ZIP**, page 6). X et Z peuvent être (A) confondus, (B) distincts ou (C) comprendre un ensemble de variables en commun.

3.4.1 Régression ZIP avec ''pscl''

Nous travaillons sur l'hypothèse (A) dans un premier scénario. Nous faisons appel à la commande `zeroInfl()` du package ''pscl'' :

```
#modélisation zero-inflated poisson regression
mzip <- zeroinfl(Affairs ~ . | .,data=DTrain)
print(summary(mzip))
```

```
Call:
zeroinfl(formula = Affairs ~ . | ., data = DTrain)

Pearson residuals:
    Min      1Q  Median      3Q     Max
-1.2247 -0.4695 -0.3686 -0.2485  9.3799

Count model coefficients (poisson with log link):
               Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)     3.149768   0.435454   7.233 4.71e-13 ***
Gender         -0.179926   0.148611  -1.211   0.2260
Age            -0.016550   0.011616  -1.425   0.1542
YearsMarried    0.049264   0.020152   2.445   0.0145 *
Children       -0.359220   0.183934  -1.953   0.0508 .
Religiousness  -0.124808   0.048542  -2.571   0.0101 *
Education       0.002755   0.032297   0.085   0.9320
Occupation      0.013678   0.041470   0.330   0.7415
RatingMarriage -0.221006   0.053844  -4.105 4.05e-05 ***

Zero-inflation model coefficients (binomial with logit link):
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.14540    1.33175  -0.109  0.91306
Gender         -0.27072    0.36667  -0.738  0.46032
Age             0.03954    0.02861   1.382  0.16697
YearsMarried   -0.07364    0.05018  -1.468  0.14222
Children       -0.40215    0.42820  -0.939  0.34765
Religiousness   0.32417    0.13124   2.470  0.01351 *
Education      -0.06422    0.07720  -0.832  0.40552
Occupation     -0.05716    0.10703  -0.534  0.59326
RatingMarriage  0.40137    0.13364   3.004  0.00267 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 26
Log-likelihood: -331.4 on 18 Df
```

Nous distinguons les deux modèles (poisson et binomial) dans la sortie de R. Ils présentent des variables pertinentes en commun et distincts si l'on se réfère à une inspection rapide.

L'objet généré par la commande possède plusieurs propriétés…

```
#propriétés de l'objet
attributes(mzip)
```

```
$names
 [1] "coefficients"  "residuals"     "fitted.values" "optim"
 [5] "method"        "control"       "start"         "weights"
 [9] "offset"        "n"             "df.null"       "df.residual"
[13] "terms"         "theta"         "SE.logtheta"   "loglik"
[17] "vcov"          "dist"          "link"          "linkinv"
[21] "converged"     "call"          "formula"       "levels"
[25] "contrasts"     "model"         "y"

$class
[1] "zeroinfl"
```

… qui permettent d'accéder à des informations supplémentaires et de réaliser des calculs intermédiaires.

Nous disposons ainsi de la log-vraisemblance (COURS ZIP, page 8) :

```
#log-vraisemblance
print(mzip$loglik)
```

```
[1] -331.4206
```

Des vecteurs des coefficients :

```
#coefficients
print(mzip$coefficients)
```

```
$count
   (Intercept)         Gender            Age   YearsMarried
   3.149767671   -0.179925857   -0.016549548    0.049263651
      Children  Religiousness      Education     Occupation
  -0.359219564   -0.124807815    0.002754745    0.013678481
RatingMarriage
  -0.221006137

$zero
   (Intercept)         Gender            Age   YearsMarried
   -0.14540349    -0.27071823     0.03954065    -0.07364180
      Children  Religiousness      Education     Occupation
   -0.40215005     0.32417378    -0.06421668    -0.05716486
RatingMarriage
    0.40137354
```

De la matrice de variance covariance des coefficients …

```
#matrice de variance covariance des coefs
print(mzip$vcov)
```

```
                    count_(Intercept)  count_Gender     count_Age count_YearsMarried count_Children count_Religiousness
count_(Intercept)        1.896203e-01  2.524192e-02 -1.252441e-03       6.174253e-04  -2.798318e-03       -5.795339e-03
count_Gender             2.524192e-02  2.208532e-02 -3.945837e-04       3.223333e-04   6.139403e-03       -6.524328e-05
count_Age               -1.252441e-03 -3.945837e-04  1.349272e-04      -1.855214e-04   1.336698e-04        1.235777e-05
count_YearsMarried       6.174253e-04  3.223333e-04 -1.855214e-04       4.060835e-04  -1.132347e-03       -1.229366e-04
count_Children          -2.798318e-03  6.139403e-03  1.336698e-04      -1.132347e-03   3.383187e-02       -1.188005e-03
count_Religiousness     -5.795339e-03 -6.524328e-05  1.235777e-05      -1.229366e-04  -1.188005e-03        2.356297e-03
count_Education         -9.314061e-03 -1.886132e-03 -6.318487e-05       1.217734e-04  -1.796854e-03        1.138474e-04
count_Occupation         8.702424e-04 -1.729656e-03  1.145184e-05       2.391071e-05  -4.474127e-04        6.004317e-06
count_RatingMarriage    -1.468925e-04  2.123022e-03 -1.142421e-04       1.853496e-04   3.993181e-03       -1.433812e-04
zero_(Intercept)         1.550746e-02  2.162986e-03 -1.445138e-04       1.237674e-04   5.911943e-05       -7.339022e-04
zero_Gender              2.036483e-03  1.698560e-03 -2.398906e-05      -1.322740e-05   9.207141e-04        5.288568e-05
zero_Age                -1.000870e-04 -1.930926e-05  1.252831e-05      -1.740785e-05   3.642653e-05        1.576189e-06
zero_YearsMarried        3.580170e-05 -3.989712e-05 -1.671400e-05       3.747046e-05  -1.880058e-04       -1.742403e-05
zero_Children           -5.644821e-05  8.978861e-04  2.884551e-05      -1.537602e-04   2.764858e-03        4.170943e-05
zero_Religiousness      -5.266639e-04  9.173486e-05 -7.054341e-07      -1.476608e-05   5.655921e-05        2.087512e-04
zero_Education          -7.794016e-04 -1.974799e-04 -7.642797e-06       1.679662e-05  -2.115807e-04        1.118104e-05
zero_Occupation          2.699298e-05 -1.011378e-04  2.510084e-06       1.446359e-06  -4.812433e-05        5.038256e-06
zero_RatingMarriage      8.854866e-05  2.710040e-04 -8.892060e-07      -9.620385e-06   5.038256e-04        3.489544e-05
                    count_Education count_Occupation count_RatingMarriage zero_(Intercept)   zero_Gender      zero_Age
count_(Intercept)     -9.314061e-03     8.702424e-04        -1.468925e-04     1.550746e-02  2.036483e-03 -1.000870e-04
count_Gender          -1.886132e-03    -1.729656e-03         2.123022e-03     2.162986e-03  1.698560e-03 -1.930926e-05
count_Age             -6.318487e-05     1.145184e-05        -1.142421e-04    -1.445138e-04 -2.398906e-05  1.252831e-05
count_YearsMarried     1.217734e-04     2.391071e-05         1.853496e-04     1.237674e-04 -1.322740e-05 -1.740785e-05
count_Children        -1.796854e-03    -4.474127e-04         3.993181e-03     5.911943e-05  9.207141e-04  3.642653e-05
count_Religiousness    1.138474e-04     6.004317e-06        -1.433812e-04    -7.339022e-04  5.288568e-05  1.576189e-06
count_Education        1.043097e-03    -5.126834e-04        -7.121525e-04    -7.450811e-04 -1.990502e-04 -1.032522e-05
count_Occupation      -5.126834e-04     1.719743e-03         1.298684e-04     1.329580e-05 -1.091451e-04  3.289723e-06
count_RatingMarriage  -7.121525e-04     1.298684e-04         2.899144e-03     9.013609e-05  3.038000e-04 -2.145853e-06
zero_(Intercept)      -7.450811e-04     1.329580e-05         9.013609e-05     1.773559e+00  1.617860e-01 -1.388639e-02
zero_Gender           -1.990502e-04    -1.091451e-04         3.038000e-04     1.617860e-01  1.344437e-01 -2.441522e-03
zero_Age              -1.032522e-05     3.289723e-06        -2.145853e-06    -1.388639e-02 -2.441522e-03  8.186047e-04
zero_YearsMarried      2.407353e-05     3.304075e-06        -1.918239e-05     1.295594e-02  2.794017e-03 -1.049633e-03
zero_Children         -2.074889e-04    -6.039477e-05         5.229509e-04    -1.155884e-01 -1.384169e-02  1.218227e-03
zero_Religiousness    -5.427466e-07    -1.026089e-05         5.498006e-05    -3.508236e-02  1.221952e-03 -7.925663e-05
zero_Education         9.171583e-05    -3.054399e-05        -7.696287e-05    -7.023843e-02 -7.130207e-03 -1.710508e-04
zero_Occupation       -3.040762e-05     1.038629e-04         2.676774e-06     6.592372e-03 -1.218093e-02 -1.618004e-04
zero_RatingMarriage   -7.954156e-05     1.613096e-06         2.466959e-04    -4.760497e-02  1.284447e-04  7.834689e-05
                    zero_YearsMarried zero_Children zero_Religiousness zero_Education zero_Occupation zero_RatingMarriage
count_(Intercept)        3.580170e-05 -5.644821e-05      -5.266639e-04  -7.794016e-04    2.699298e-05        8.854866e-05
count_Gender            -3.989712e-05  8.978861e-04       9.173486e-05  -1.974799e-04   -1.011378e-04        2.710040e-04
count_Age               -1.671400e-05  2.884551e-05      -7.054341e-07  -7.642797e-06    2.510084e-06       -8.892060e-07
count_YearsMarried       3.747046e-05 -1.537602e-04      -1.476608e-05   1.679662e-05    1.446359e-06       -9.620385e-06
count_Children          -1.880058e-04  2.764858e-03       5.655921e-05  -2.115807e-04   -4.812433e-05        5.038256e-04
count_Religiousness     -1.742403e-05  4.170943e-05       2.087512e-04   1.118104e-05   -6.220318e-06        3.489544e-05
count_Education          2.407353e-05 -2.074889e-04      -5.427466e-07   9.171583e-05   -3.040762e-05       -7.954156e-05
count_Occupation         3.304075e-06 -6.039477e-05      -1.026089e-05  -3.054399e-05    1.038629e-04        1.613096e-06
count_RatingMarriage    -1.918239e-05  5.229509e-04       5.498006e-05  -7.696287e-05    2.676774e-06        2.466959e-04
zero_(Intercept)         1.295594e-02 -1.155884e-01      -3.508236e-02  -7.023843e-02    6.592372e-03       -4.760497e-02
zero_Gender              2.794017e-03 -1.384169e-02       1.221952e-03  -7.130207e-03   -1.218093e-02        1.284447e-04
```

```
zero_Age              -1.049633e-03  1.218227e-03      -7.925663e-05  -1.710508e-04   -1.618004e-04        7.834689e-05
zero_YearsMarried      2.517912e-03 -8.523350e-03      -1.009247e-03   2.869943e-04    8.097975e-05        7.170183e-04
zero_Children         -8.523350e-03  1.833568e-01      -1.415783e-03  -1.266846e-03    4.951161e-03        4.248409e-03
zero_Religiousness    -1.009247e-03 -1.415783e-03       1.722480e-02   1.847669e-04   -5.559211e-04       -8.479040e-04
zero_Education         2.869943e-04 -1.266846e-03       1.847669e-04   5.960217e-03   -3.106070e-03       -1.844463e-03
zero_Occupation        8.097975e-05  4.951161e-03      -5.559211e-04  -3.106070e-03    1.145469e-02        6.054807e-04
zero_RatingMarriage    7.170183e-04  4.248409e-03      -8.479040e-04  -1.844463e-03    6.054807e-04        1.785837e-02
```

… pas très facile à lire, il faut le reconnaître.

3.4.2 Pertinence du modèle logistique

Dans cette section, nous évaluons la pertinence de la partie logistique du modèle en retirant toutes les variables. Attention, notre modèle n'est pas équivalent à la régression de Poisson dans ce cas puisque nous estimons quand-même la probabilité structurelle π_i d'obtenir la valeur ''0'', mais à l'aide d'une constante c.-à-d. nous attribuons la même probabilité à tous les individus (**COURS ZIP**, page 13).

Ce modèle s'écrit comme suit sous R :

```
#approximation de pi_i par une constante
mzip_ref <- zeroinfl(Affairs ~ . | 1,data=DTrain)
print(summary(mzip_ref))
```

```
Call:
zeroinfl(formula = Affairs ~ . | 1, data = DTrain)

Pearson residuals:
    Min      1Q  Median      3Q     Max
-0.5093 -0.4759 -0.4587 -0.4335  7.8481

Count model coefficients (poisson with log link):
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     3.150391   0.438947   7.177 7.12e-13 ***
Gender         -0.185142   0.150606  -1.229  0.21895
Age            -0.017474   0.011760  -1.486  0.13731
YearsMarried    0.051545   0.020354   2.532  0.01133 *
Children       -0.370771   0.187469  -1.978  0.04795 *
Religiousness  -0.132041   0.049331  -2.677  0.00744 **
Education       0.006279   0.032901   0.191  0.84865
Occupation      0.014133   0.042037   0.336  0.73671
RatingMarriage -0.229124   0.054915  -4.172 3.01e-05 ***

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.2559     0.1416   8.867   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 18
Log-likelihood: -344.2 on 10 Df
```

Le modèle est en deux parties toujours. Mais la fraction binomiale est réduite à la constante.

La probabilité pour n'importe quel individu d'être fidèle avec ce modèle est calculée en deux temps (**COURS ZIP**, pages 13 et 16). Nous calculons λ_i :

$$\hat{\lambda}_i = \exp \hat{a}_0 = \exp 1.2559 = 3.511064, \forall i$$

```
#lambda
lambda_i <- exp(mzip_ref$coefficients$zero[1])
print(lambda_i)
```

```
3.511064
```

Puis la probabilité π_i :

$$\hat{\pi}_i = \frac{\hat{\lambda}_i}{1 + \hat{\lambda}_i} = 0.7783228, \forall i$$

```
#probabilité de fidélité
pi_i <- lambda_i/(1 + lambda_i)
print(pi_i)
```

```
0.7783228
```

C'est plutôt rassurant (*je ne sais pas…*), en dehors de toute considération sur les caractéristiques des personnes, la probabilité d'être fidèle est de 77%. Alors que la proportion de valeurs ''0'' sur le même ensemble de données est …

```
#proportion de valeurs 0 dans TRAIN
print(table(DTrain$Affairs)/nrow(DTrain))
```

```
        0         1         2         3         7        12
0.78405316 0.05647841 0.01993355 0.02657807 0.06644518 0.04651163
```

… de 78.4%. Cela accrédite l'idée qu'une partie des personnes qui n'ont pas fauté sont des libertins qui n'ont pas eu (encore) l'occasion de sévir.

Nous réalisons un test du rapport de vraisemblance pour vérifier l'hypothèse de nullité de tous les coefficients hors constante du modèle binomial (**COURS ZIP**, page 13). Nous opposons les log-vraisemblance des deux modèles à opposer (sur la partie LOGIT, modèle réduit à la constante vs. modèle complet) :

```
#test du rapport de vraisemblance
LR_Ref <- 2*(mzip$loglik - mzip_ref$loglik)
print(LR_Ref)
```

```
25.63115
```

Avec pour probabilité critique :

```
#p-value
print(pchisq(LR_Ref,df=8,lower.tail=FALSE))
```

```
0.001214282
```

Au risque 5%, l'hypothèse nulle n'est pas accréditée par les données. Il existe des explicatives pertinentes pour expliquer la valeur structurelle '' 0'' de la variable ''Affairs''.

3.4.3 Sélection de variables

La sélection automatique de variables est particulière ardue dans le contexte de la régression ZIP parce que les modèles sont enchevêtrés. Le package d'ailleurs ne propose pas d'outils en ce sens. En me basant sur les coefficients significatifs dans les deux premières approches (logistique et Poisson), en tâtonnant un peu aussi, j'ai fini par définir un second modèle.

```
#modèle simplifié
mzips <- zeroinfl(Affairs ~ YearsMarried | Religiousness + RatingMarriage,data=DTrain)
print(summary(mzips))
```

```
Call:
zeroinfl(formula = Affairs ~ YearsMarried | Religiousness + RatingMarriage, data =
DTrain)

Pearson residuals:
    Min      1Q  Median      3Q     Max
-0.9704 -0.4865 -0.3903 -0.2773  5.1969

Count model coefficients (poisson with log link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.51819    0.11288  13.450   <2e-16 ***
YearsMarried  0.02071    0.01037   1.996   0.0459 *

Zero-inflation model coefficients (binomial with logit link):
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.2816     0.6065  -2.113 0.034582 *
Religiousness    0.2918     0.1239   2.355 0.018515 *
RatingMarriage   0.4453     0.1233   3.612 0.000304 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 12
Log-likelihood: -356.2 on 5 Df
```

Nous expliquons :

- ''Fidélité vs. adultère'' (binomial) par ''Religiousness'' et ''RatingMarriage'', toutes les deux ont une influence positive sur la fidélité. Moralité, prions pour être heureux en ménage.
- ''Le nombre de liaisons'' (count) par les années de mariage. Le coefficient est positif. Avec le temps, tout fout le camp, on se met à regarder ailleurs pour voir si l'herbe n'y serait pas plus verte. Extirpant de plus profond de l'âme de l'être désespéré des chansons poignantes.

De nouveau, nous pourrions réaliser un test du rapport de vraisemblance pour vérifier la nullité effective des coefficients des variables qui ont été éliminées. Mais je préfère passer par le test de Vuong qui a un champ d'application plus large et permet de comparer les performances prédictives des modèles.

3.5 Test de Vuong pour la comparaison de modèles

J'ai découvert le test de Vuong en lisant la documentation du package ''pscl'' qui en propose une implémentation. Il oppose les performances prédictives des modèles, qu'ils soient imbriqués ou non. Il est plus générique que le test usuel du rapport de vraisemblance qui n'est fonctionnel que pour les modèles imbriqués.

3.5.1 Régression ZIP vs. Régression de Poisson

Le premier propos du test de Vuong est de comparer les mérites respectifs des modèles ''zero-inflated Poisson'' et Poisson classique. Il est mis en avant en ce sens dans la documentation du package et dans les tutoriels que nous trouvons en ligne. Il est basé sur la propension des modèles à identifier la valeur adéquate de la cible $\hat{P}_{modèle}(Y = y_i)$ (**COURS ZIP**, page 14).

Pour un échantillon de taille n_{train}, si nous opposons les modèles M1 (ex. Régression ZIP avec sélection de variables, section 3.4.3) et M2 (ex. Régression de Poisson avec sélection de variables, section 3.3), nous formons :

$$v_i = \ln \hat{P}_{M1}(Y = y_i) - \ln \hat{P}_{M2}(Y = y_i)$$

Puis la statistique de test :

$$V = \sqrt{n_{train}} \frac{\bar{v}}{\sigma_v}$$

Où $\bar{v}$ et σ_v sont respectivement la moyenne et l'écart-type de (v_i). Elle suit approximativement une distribution normale centrée et réduite.

Le rôle de $\hat{P}_{modèle}(Y = y_i)$ est central dans ce calcul, elle représente la probabilité d'obtenir la bonne valeur de la cible avec les paramètres estimés de (π_i) et (μ_i) fournis par la combinaison de modèles pour l'individu n°i (**COURS ZIP**, page 16). Elle nous est fournie par la fonction `predict()`. Appliquée sur l'échantillon d'apprentissage, nous disposons à la sortie…

```
#structure de la prédiction
pzip <- predict(mzips,newdata=DTrain,type="prob")
print(dim(pzip))
```

```
301  13
```

… d'une matrice (301 x 13) parce notre échantillon est composé de 301 observations, et que la plage des valeurs possibles de ''Affairs'' va de 0 à 12 (Figure 1).

Affichons les 3 premières lignes de pzip.

```
#affichage des 3 premières lignes
print(head(pzip,3))
```

```
          0           1           2           3          4          5          6
1 0.8922729 0.001328465 0.004135740 0.008583516 0.01336098 0.01663802 0.01726568
2 0.8417127 0.003252273 0.009129079 0.017083457 0.02397650 0.02692067 0.02518863
3 0.7771410 0.010120912 0.023457285 0.036244707 0.04200226 0.03893954 0.03008345
           7          8           9          10          11           12
1 0.01535744 0.01195259 0.008268995 0.005148562 0.002914248 0.0015120928
2 0.02020117 0.01417609 0.008842684 0.004964255 0.002533562 0.0011852780
3 0.01992130 0.01154292 0.005945131 0.002755812 0.001161301 0.0004485923
```

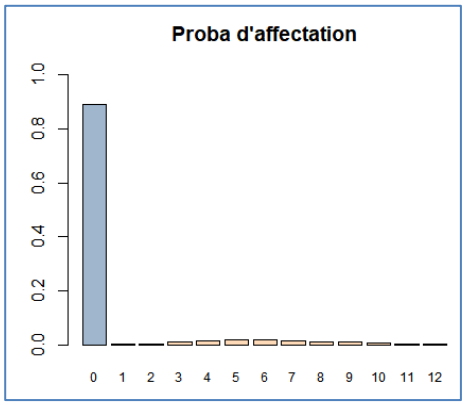
Pour l'individu n°1, P_{MZIPS}(Affairs == 0) = 0.8922729, P_{MZIPS}(Affairs == 1) = 0.001328465, …, et P_{MZIPS}(Affairs == 12) = 0.0015120928. La somme en ligne de la matrice est forcément égale à 1. Et la bonne valeur de la classe pour cet individu est $y_1 = 0$.

Nous créons une fonction pour afficher ces probabilités sous forme de diagramme en bâtons. La bonne étiquette est soulignée avec une couleur différente.

```
#diagramme en bâton
diag_baton <- function(data,proba,i){
  #plage de valeurs
  plage <- seq(min(data$Affairs),max(data$Affairs),1)
  #couleur
  couleur <- ifelse(DTrain$Affairs[i]==plage,"slategray3","peachpuff")
  #barplot
  barplot(proba[i,],col=couleur,ylim=c(0,1),main="Proba d'affectation",cex.names=0.75)
}
```

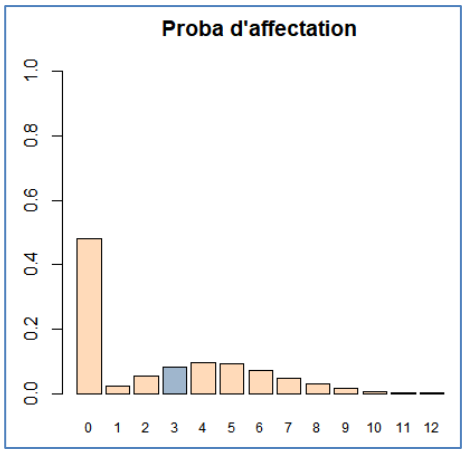
Ainsi, pour l'individu n°1 dont la vraie étiquette est $y_1 = 0$, nous avons la distribution :

```
#individu n_1 du TRAIN - classe 0
diag_baton(DTrain,pzip,1)
```



Pour l'individu n°241 avec $y_{241} = 3$, le modèle fournit :

```
#individu n_241 du TRAIN - classe 3
diag_baton(DTrain,pzip,241)
```



Le test de Vuong oppose les ''bonnes'' probabilités (la barre bleue) des modèles pour déterminer celui qui est le plus performant. Si (V > 0), le premier modèle est supérieur au second, (V < 0) dans le cas inverse.

Revenons à la confrontation entre MZIPS (Régression ZIP avec sélection de variables) et MPRS (Régression de Poisson avec sélection itou) en faisant appel à la fonction vuong() :

```
#test Vuong - ZIP simplifié vs. Poisson simplifié
pscl::vuong(mzips,mprs)
```

```
Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
 null that the models are indistinguishible)
-------------------------------------------------------------
              Vuong z-statistic             H_A    p-value
Raw                   5.910573 model1 > model2 1.7046e-09
AIC-corrected         5.931473 model1 > model2 1.5011e-09
BIC-corrected         5.970210 model1 > model2 1.1847e-09
```

Manifestement, pour notre ensemble de données, la régression ZIP (MZIPS) est largement supérieure à la régression de Poisson.

Le package ''pscl'' propose deux mesures corrigées. L'ajustement dépend de la complexité des modèles et des effectifs. Posons K_1 et K_2 le nombre de paramètres estimés (incluant les constantes) dans les deux modèles M1 et M2 à opposer, nous avons (dixit le code source de ''pscl'', voir le fichier ''**vuong.R**'') :

- Pour la correction AIC :

$$v_i^{AIC} = \left[\ln \hat{P}_{M1}(Y = y_i) - \ln \hat{P}_{M2}(Y = y_i)\right] - \frac{(K_1 - K_2)}{n_{train}}$$

- Pour la correction BIC :

$$v_i^{BIC} = \left[\ln \hat{P}_{M1}(Y = y_i) - \ln \hat{P}_{M2}(Y = y_i)\right] - \frac{(K_1 - K_2)}{2 \times n_{train}} \times \ln n_{train}$$

Les statistiques V^{AIC} et V^{BIC} sont formées de la même manière que précédemment.

Deux remarques : (1) la correction est minime lorsque les effectifs sont élevés, (2) les indicateurs peuvent aboutir à des conclusions contradictoires.

3.5.2 Modèle complet vs. Modèle simplifié

Puisque le test de Vuong s'applique pour les modèles non-imbriqués ou imbriqués, nous le mettons en œuvre pour vérifier l'intérêt de la sélection de variables dans la régression ZIP. Nous avions conservé 2 variables (au lieu de 8) dans la partie LOGIT, et 1 variable explicative (au lieu de 8 toujours) dans la partie POISSON. Était-ce à bon escient ?

```
#test Vuong - ZIP simplifié (model 1) vs. ZIP complet (model 2)
pscl::vuong(mzips,mzip)
```

```
Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
 null that the models are indistinguishible)
-------------------------------------------------------------
              Vuong z-statistic             H_A   p-value
Raw                  -2.351276 model2 > model1 0.0093546
AIC-corrected        -1.117590 model2 > model1 0.1318710
BIC-corrected         1.169115 model1 > model2 0.1211787
```

Les résultats ne sont pas tranchés. Pour la statistique non-corrigée (V), le modèle complet (MZIP, model 2) semble meilleur à 5%. Ce que réfute les autres indicateurs (V^{AIC} et V^{BIC}) qui n'identifient pas une différence significative.

3.5.3 Test de Vuong à partir d'un échantillon test

Cette apparente contradiction m'a amené à redéfinir le test de Vuong en l'appliquant sur un échantillon test, un ensemble qui n'a pas participé à l'élaboration du modèle. En effet, le test non-corrigé fait la part belle au surapprentissage lorsqu'il est calculé sur l'échantillon qui a servi à la construction des modèles. L'introduction d'une correction basée sur leur complexité permet de remédier à ce biais. Une autre approche – pour la même finalité – consiste à utiliser un échantillon à part, une sorte d'arbitre impartial qui permet de juger objectivement les performances prédictives des modèles. Dans ce cas, le test non corrigé calculé sur un échantillon distinct devrait être cohérent avec le test corrigé calculé sur l'échantillon d'apprentissage.

La fonction vuong() du package ''pscl'' n'a pas prévu de travailler sur un échantillon distinct. Je l'ai donc réécrite de manière très succincte en introduisant un paramètre supplémentaire ''data'' :

```
#ré-écrire le test de Vuong
my_vuong <- function(m1,m2,data){
```

```
  #prédiction des probabilités
  pp1 <- predict(m1,newdata=data,type="prob")
  pp2 <- predict(m2,newdata=data,type="prob")
  #probabilités
  p1 <- sapply(1:nrow(data),function(i){pp1[i,data$Affairs[i]+1]})
  p2 <- sapply(1:nrow(data),function(i){pp2[i,data$Affairs[i]+1]})
  #écart
  v <- log(p1) - log(p2)
  #stat de test
  vuong <- sqrt(nrow(data))*mean(v)/sd(v)
  #p-value
  pvalue <- ifelse(vuong > 0,pnorm(vuong,lower.tail=FALSE),pnorm(vuong,lower.tail=TRUE))
  #
  return(list(Vuong=vuong,Pvalue=pvalue))
}
```

Appliqué sur l'échantillon d'apprentissage (DTrain)…

```
#refaire l'évaluation sur TRAIN
print(my_vuong(mzips,mzip,data=DTrain))
```

```
$Vuong
[1] -2.351276

$Pvalue
[1] 0.00935457
```

… nous retrouvons à l'identique la statistique non-corrigée ci-dessus (heureusement !).

Appliqué sur l'échantillon test (DTest) maintenant…

```
#la même évaluation sur TEST
print(my_vuong(mzips,mzip,data=DTest))
```

```
$Vuong
[1] 0.3792208

$Pvalue
[1] 0.352262
```

… la procédure nous confirme que les deux modèles présentent des performances prédictives similaires. Nous appliquons le principe de parcimonie dans ce cas, nous préférons le modèle simplifié.

Nous constatons surtout que le travail sur un échantillon test nous affranchit de la prise en compte de la complexité. Si l'on regarde un peu plus loin, il nous permet même de confronter des modèles reposant sur des systèmes de représentation différents.

4 Traitements sous Python avec ''statsmodels''

Dans cette section, nous reprenons sous Python les traitements ci-dessus. Nous utilisons le package ''statsmodels'' que nous avions déjà étudié précédemment (Septembre 2015). Nous irons moins dans le détail des résultats puisque nous les avions commentés déjà. L'objectif est surtout de montrer les équivalences entre les deux outils.

4.1 Importation et préparation des données

De nouveau, nous importons les données.

```
#modifier le dossier de travail
import os
os.chdir("... votre dossier ...")

#importer les données
import pandas
D = pandas.read_excel("infidelite_zip_reg.xlsx",sheet_name=0)
print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 601 entries, 0 to 600
Data columns (total 9 columns):
Affairs           601 non-null int64
Gender            601 non-null int64
Age               601 non-null float64
YearsMarried      601 non-null float64
Children          601 non-null int64
Religiousness     601 non-null int64
Education         601 non-null int64
Occupation        601 non-null int64
RatingMarriage    601 non-null int64
dtypes: float64(2), int64(7)
memory usage: 42.3 KB
```

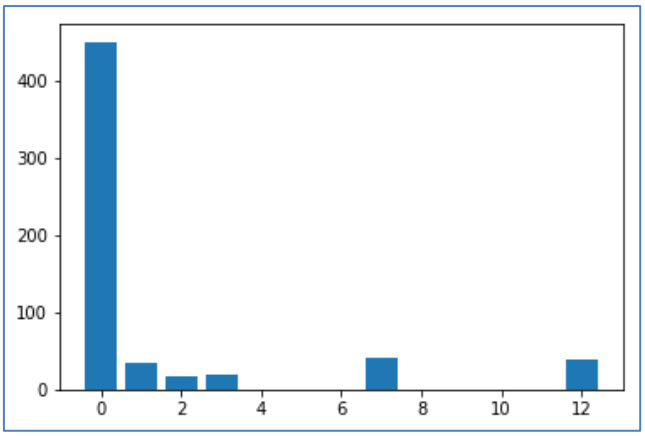
Nous comptabilisons les valeurs de la variable cible ''Affairs''.

```
#comptage des valeurs
import numpy
nbAffairs = numpy.unique(D.Affairs,return_counts=True)
print(nbAffairs)
```

```
(array([ 0,  1,  2,  3,  7, 12], dtype=int64), array([451,  34,  17,  19,  42,
38], dtype=int64))
```

Avec un petit `barplot()` pour représenter les effectifs.

```
#barplot
import matplotlib.pyplot as plt
plt.bar(nbAffairs[0],nbAffairs[1])
```



Nous partitionnons les données en échantillons d'apprentissage et de test, avec exactement la même procédure que sous R (les 301 premiers en apprentissage, les 300 suivants en test).

```
#séparer y (vecteur de la var. cible) et X (matrice des explicatives)
y = D.iloc[:,0]
X = D.iloc[:,1:]

#partition des données en apprentissage test
yTrain = y[:301]
yTest = y[301:]

XTrain = X.iloc[:301,:]
XTest = X.iloc[301:,:]

#dénombrement en TRAIN
numpy.unique(yTrain,return_counts=True)
```

```
(array([ 0,  1,  2,  3,  7, 12], dtype=int64),
 array([236,  17,   6,   8,  20,  14], dtype=int64))
```

4.2 Régression logistique

Pour réaliser la régression logistique, nous codons au préalable la variable dépendante.

```
#codage binaire
YBin = numpy.zeros(yTrain.shape)
YBin[yTrain==0] = 1

#vérification
print(numpy.sum(YBin))
```

```
236.0
```

Singularité du package ''statsmodels'', nous devons rajouter explicitement la constante dans la matrice des explicatives.

```
#ajouter la constante au XTrain
import statsmodels
XTrainConst = statsmodels.tools.add_constant(XTrain)
```

Nous pouvons ensuite pratiquer la régression logistique. Les données (YBin et XTrainsConst) sont transmises lors de l'instanciation de la classe de calcul.

```
#régression logistique
from statsmodels.discrete.discrete_model import Logit
mlr = Logit(YBin,XTrainConst)
res_mlr = mlr.fit()

#affichage
print(res_mlr.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.477088
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  301
Model:                          Logit   Df Residuals:                      292
Method:                           MLE   Df Model:                            8
Date:                Mon, 10 Jun 2019   Pseudo R-squ.:                 0.08556
Time:                        18:02:15   Log-Likelihood:                -143.60
converged:                       True   LL-Null:                       -157.04
                                        LLR p-value:                 0.0007432
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             -0.2044      1.325     -0.154      0.877      -2.802       2.393
Gender            -0.2552      0.365     -0.698      0.485      -0.971       0.461
Age                0.0406      0.028      1.427      0.154      -0.015       0.096
YearsMarried      -0.0773      0.050     -1.550      0.121      -0.175       0.020
Children          -0.3701      0.426     -0.869      0.385      -1.205       0.465
Religiousness      0.3326      0.131      2.547      0.011       0.077       0.589
Education         -0.0655      0.077     -0.852      0.394      -0.216       0.085
Occupation        -0.0578      0.107     -0.541      0.588      -0.267       0.151
RatingMarriage     0.4130      0.133      3.103      0.002       0.152       0.674
==============================================================================
```

Les résultats concordent en tous points avec ceux de R (section 3.2).

Ici également, l'objet résultat présente des propriétés et méthodes exploitables pour la réalisation de calculs supplémentaires.

```
#propriétés
print(dir(res_mlr))
```

```
['__class__',    '__delattr__',    '__dict__',    '__dir__',    '__doc__',    '__eq__',
'__format__',  '__ge__',  '__getattribute__',  '__getstate__',  '__gt__',  '__hash__',
'__init__',   '__init_subclass__',   '__le__',   '__lt__',   '__module__',   '__ne__',
'__new__',  '__reduce__',  '__reduce_ex__',  '__repr__',  '__setattr__',  '__sizeof__',
'__str__',     '__subclasshook__',     '__weakref__',     '_cache',     '_data_attr',
'_get_endog_name',   '_get_robustcov_results',   'aic',   'bic',   'bse',   'conf_int',
```

```
'cov_kwds',    'cov_params',    'cov_type',    'df_model',    'df_resid',    'f_test',
'fittedvalues', 'get_margeff', 'initialize', 'k_constant', 'llf', 'llnull', 'llr',
'llr_pvalue',    'load',    'mle_retvals',    'mle_settings',    'model',    'nobs',
'normalized_cov_params', 'params', 'pred_table', 'predict', 'prsquared', 'pvalues',
'remove_data', 'resid_dev', 'resid_generalized', 'resid_pearson', 'resid_response',
'save',    'scale',    'set_null_options',    'summary',    'summary2',    't_test',
't_test_pairwise', 'tvalues', 'use_t', 'wald_test', 'wald_test_terms']
```

Pour accéder directement aux coefficients estimés par exemple :

```
#accès aux coefficients estimés
print(res_mlr.params)
```

```
const            -0.204424
Gender           -0.255198
Age               0.040579
YearsMarried     -0.077273
Children         -0.370142
Religiousness     0.332606
Education        -0.065454
Occupation       -0.057830
RatingMarriage    0.413016
dtype: float64
```

Pour dériver la déviance à partir de la log-vraisemblance :

```
#residual deviance
print("Residual deviance =",-2*res_mlr.llf)
```

```
Residual deviance = 287.20716986226904
```

Ou encore pour calculer la statistique du test de rapport de vraisemblance de significativité globale du modèle :

```
#stat. de test de vraisemblance - signif. globale
print("LR (signif.globale) =",res_mlr.llr)
```

```
LR (signif.globale) = 26.874252460555738
```

Et la probabilité critique associée :

```
#p-value associée
print("p-value =",res_mlr.llr_pvalue)
```

```
p-value = 0.0007432108837191402
```

4.3 Régression de Poisson

Nous faisons appel à la classe `Poisson` pour la régression de Poisson.

```
#régression de Poisson
from statsmodels.discrete.discrete_model import Poisson
mpr = Poisson(yTrain,XTrainConst)
res_mpr = mpr.fit()

#affichage
```

```
print(res_mpr.summary())
```

```
Optimization terminated successfully.
         Current function value: 2.118537
         Iterations 6
                          Poisson Regression Results
==============================================================================
Dep. Variable:                Affairs   No. Observations:                  301
Model:                        Poisson   Df Residuals:                      292
Method:                           MLE   Df Model:                            8
Date:                Mon, 10 Jun 2019   Pseudo R-squ.:                  0.1656
Time:                        18:29:45   Log-Likelihood:                -637.68
converged:                       True   LL-Null:                       -764.20
                                        LLR p-value:                3.909e-50
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              2.9925      0.457      6.553      0.000       2.097       3.888
Gender             0.1525      0.135      1.127      0.260      -0.113       0.418
Age               -0.0491      0.011     -4.451      0.000      -0.071      -0.027
YearsMarried       0.0988      0.018      5.396      0.000       0.063       0.135
Children           0.0488      0.167      0.293      0.770      -0.278       0.376
Religiousness     -0.3476      0.048     -7.179      0.000      -0.442      -0.253
Education          0.0196      0.028      0.708      0.479      -0.035       0.074
Occupation         0.0421      0.038      1.122      0.262      -0.031       0.116
RatingMarriage    -0.4754      0.045    -10.500      0.000      -0.564      -0.387
==============================================================================
```

Sans surprise aucune, les résultats sont identiques à ceux de R.

4.4 Régression ZIP

4.4.1 Régression complète

Pour la régression ZIP, nous faisons appel à la classe `ZeroInflatedPoisson`. Outre le vecteur cible (yTrain), deux blocs d'explicatives sont passés à l'instanciation : le premier pour la partie Poisson, la seconde pour la partie Logistique. Nous utilisons le même ensemble de variables dans cette section.

```
#ZIP regression
from statsmodels.discrete.count_model import ZeroInflatedPoisson
mzip = ZeroInflatedPoisson(yTrain,XTrainConst,XTrainConst)
res_mzip = mzip.fit(maxiter=100)

#summary
print(res_mzip.summary())
```

```
Optimization terminated successfully.
         Current function value: 1.101065
         Iterations: 53
         Function evaluations: 60
         Gradient evaluations: 60
                     ZeroInflatedPoisson Regression Results
==============================================================================
```

```
Dep. Variable:                  Affairs   No. Observations:                  301
Model:             ZeroInflatedPoisson   Df Residuals:                      292
Method:                            MLE   Df Model:                            8
Date:                 Mon, 10 Jun 2019   Pseudo R-squ.:                 0.09860
Time:                         18:33:46   Log-Likelihood:                -331.42
converged:                        True   LL-Null:                       -367.67
                                         LLR p-value:                 1.558e-12
==============================================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
inflate_const                -0.1464      1.331     -0.110      0.912      -2.755       2.463
inflate_Gender               -0.2708      0.366     -0.739      0.460      -0.989       0.447
inflate_Age                   0.0395      0.029      1.383      0.167      -0.016       0.096
inflate_YearsMarried         -0.0736      0.050     -1.469      0.142      -0.172       0.025
inflate_Children             -0.4022      0.428     -0.940      0.347      -1.241       0.436
inflate_Religiousness         0.3242      0.131      2.472      0.013       0.067       0.581
inflate_Education            -0.0642      0.077     -0.832      0.406      -0.215       0.087
inflate_Occupation           -0.0572      0.107     -0.534      0.593      -0.267       0.153
inflate_RatingMarriage        0.4014      0.134      3.006      0.003       0.140       0.663
const                         3.1497      0.435      7.236      0.000       2.297       4.003
Gender                       -0.1799      0.148     -1.212      0.226      -0.471       0.111
Age                          -0.0165      0.012     -1.425      0.154      -0.039       0.006
YearsMarried                  0.0493      0.020      2.447      0.014       0.010       0.089
Children                     -0.3592      0.184     -1.955      0.051      -0.719       0.001
Religiousness                -0.1248      0.049     -2.573      0.010      -0.220      -0.030
Education                     0.0028      0.032      0.085      0.932      -0.060       0.066
Occupation                    0.0137      0.041      0.330      0.741      -0.068       0.095
RatingMarriage               -0.2210      0.054     -4.111      0.000      -0.326      -0.116
==============================================================================================
D:\Logiciels\Anaconda3\lib\site-packages\statsmodels\base\model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

Bien que nous ayons les bons résultats (section 3.4.1), Python nous envoie un message d'avertissement indiquant l'absence de convergence du processus d'optimisation de la log-vraisemblance. Pour les vérifier, il nous enjoint à utiliser la propriété mle_retvals. Je m'empresse de le faire.

```
#inspecter l'optimisation
print(res_mzip.mle_retvals)
```

```
{'fopt': 1.1010650054986841, 'gopt': array([-1.95466036e-06,  6.90304056e-07,
5.55036896e-06,  1.45961343e-06,
       -1.41433870e-06, -8.35605990e-08,  3.52096421e-06, -2.34529206e-07,
       -1.25897188e-06, -1.43601629e-07, -1.45827830e-06, -1.63749670e-06,
       -8.18038133e-07,  5.81378366e-07, -3.42892744e-07, -1.07272015e-06,
       -2.41120211e-07,  -2.76124395e-07]), 'Hinv': array([[ 4.23824194e+02,
3.86443577e+01, -3.35094542e+00,
        3.15599750e+00, -2.84316773e+01, -8.30243789e+00,
       -1.66498320e+01,  1.52956458e+00, -1.16334944e+01,
...
       -4.50290654e-02,  5.50475488e-01, -2.93758584e-02,
        6.45437235e-02,  1.01494666e+00, -7.63920382e-02,
       -2.44019939e-01,  9.08453043e-02,  9.49497628e-01]]), 'fcalls': 60,
'gcalls': 60, 'warnflag': 0, 'converged': True}
```

Elle fournit la valeur de la fonction objectif (fopt), le vecteur gradient (gopt) et l'inverse de la matrice hessienne (Hinv) (**COURS ZIP**, page 9). Le vecteur ''gopt'' nous intéresse en particulier. A l'optimum, toutes ses valeurs doivent être nulles, et c'est le cas ici. De plus, le flag ''converged'' est bien passé à True. Le warning était une fausse alerte donc.

Nous ne le ferons pas dans ce tutoriel, mais ça peut être un exercice intéressant, nous pouvons reproduire le test de Vuong en exploitant la matrice des probabilités de prédiction qui présente les mêmes caractéristiques que sous R (section 3.5.1).

```
#probabilités d'affectation aux valeurs
pzip = mzip.predict(res_mzip.params,XTrainConst,which="prob")
print(pzip.shape)
```

```
(301, 13)
```

4.4.2 Régression avec des sous-ensembles différenciés de variables

Nous utilisons des sous-ensembles de variables dissociés pour la partie LOGIT (ZTrainSimp) et POISSON (XTrainSimp). Notre objectif est de reproduire le modèle avec sélection de variables plus haut (section 3.4.3).

```
#data pour ZIP Regression simplifié
XTrainSimp = XTrainConst.iloc[:,[0,3]]
ZTrainSimp = XTrainConst.iloc[:,[0,5,8]]

#ZIP simplifié
mzips = ZeroInflatedPoisson(yTrain,XTrainSimp,ZTrainSimp)
res_mzips = mzips.fit(maxiter=100,tol=0.00001)

#affichage
print(res_mzips.summary())
```

```
Optimization terminated successfully.
         Current function value: 1.183379
         Iterations: 20
         Function evaluations: 24
         Gradient evaluations: 24
                     ZeroInflatedPoisson Regression Results
==============================================================================
Dep. Variable:                 Affairs   No. Observations:                  301
Model:             ZeroInflatedPoisson   Df Residuals:                      299
Method:                            MLE   Df Model:                            1
Date:                Mon, 10 Jun 2019   Pseudo R-squ.:                 0.03121
Time:                        18:47:45   Log-Likelihood:                -356.20
converged:                       True   LL-Null:                       -367.67
                                        LLR p-value:                 1.663e-06
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
inflate_const     -1.2814      0.606     -2.113      0.035      -2.470      -0.093
```

```
inflate_Religiousness      0.2918      0.124      2.355      0.019      0.049      0.535
inflate_RatingMarriage     0.4452      0.123      3.611      0.000      0.204      0.687
const                      1.5182      0.113     13.452      0.000      1.297      1.739
YearsMarried               0.0207      0.010      1.996      0.046      0.000      0.041
================================================================================================
```

Faut-il le préciser encore une fois, nous retrouvons (avec de très légères différences quand-même à partir de la 4ème décimale) les résultats du package ''pscl'' pour R (section 3.4.3).

5 Conclusion

La régression ZIP est une extension de la régression de Poisson pour les configurations où la variable cible comporte une surreprésentation de la valeur ''0''. Dans ce tutoriel, nous avons vu plusieurs facettes de leur mise en œuvre sous R et sous Python, avec respectivement les packages ''pscl'' et ''statsmodels''. Dans les grandes lignes, les modes opératoires et les résultats des deux outils se rejoignent. Cette étude aura aussi été pour moi l'occasion d'explorer un peu le test de Vuong de comparaison de modèles en écrivant une variante applicable sur un échantillon test distinct des données de modélisation.

6 Références

[**COURS POISSON**] R. Rakotomalala, ''Régression de Poisson – Diapos'', mai 2019.

[**COURS ZIP**] R. Rakotomalala, ''Régression ZIP – Diapos'', juin 2019.

Statsmodels : Statistics in Python -- https://www.statsmodels.org/stable/index.html

pscl : Political Science Computational Library -- https://cran.r-project.org/package=pscl