Examen du cours d'introduction à la statistique et à l'analyse de données

Christophe Pouzat

Jeudi 12 juin 2014

1 Problème 1 : familles de 12 enfants (7 points)

À la fin du XIX siècle une étude portant sur 6115 familles de 12 enfants (les 12 enfants de chacune des familles étant issus du même père et de la même mère) a donné la répartition suivante pour le nombre de filles :

Nombre de filles	0	1	2	3	4	5	6	7	8	9	10	11	12
Nombre de familles	7	45	181	478	829	1112	1343	1033	670	286	104	24	3

Question 1.1 Si les sexes des enfants successifs d'une famille sont indépendants et que la probabilité θ_i d'avoir une fille, pour une famille i donnée, demeure constante au cours du temps, le nombre de filles X_i dans une famille de 12 enfants doit suivre une loi binomiale :

$$\Pr\{X_i = x_i\} = \frac{12!}{(12-x_i)!\,x_i!}\,\theta_i^{x_i}\,(1-\theta_i)^{12-x_i}, \quad 0 \leq \theta_i \leq 1 \text{ et } x_i = 0, 1, \ldots, 12\,.$$

Si nous supposons que θ_i a la même valeur θ pour chacune des 6115 familles de l'étude, alors la table ci-dessus doit représenter le résultat de 6115 tirages IID suivant une loi binomiale ayant une probabilité de succès θ et 12 essais. Vous écrirez la vraisemblance de θ et vous obtiendrez (en me donnant quelques détails) l'estimateur du maximum de vraisemblance (EMV) $\hat{\theta}$ de θ.

Question 1.2 Quand vous aurez obtenu $\hat{\theta}$ vous calculerez la valeur prédite du nombre de filles (parmi 12 enfants) lorsque 6115 familles sont considérées, c'est-à-dire que vous calculerez :

$$\hat{n}_i \doteq 6115\,\frac{12!}{(12-i)!\,i!}\,\hat{\theta}^i\,(1-\hat{\theta})^{12-i}, \quad \text{pour} \quad i = 0, 1, \ldots, 12\,.$$

Vous ferez un graphe de $n_i - \hat{n}_i$ en fonction de i (où les n_i sont les nombres de la seconde ligne de la table ci-dessus). Commentez.

Aide La fonction `factorial` du module `scipy.misc` calcule les factoriels. Vous pouvez y accèder avec :

```
from scipy.misc import factorial
```

Pour avoir de l'aide, vous pouvez (avec `IPython`) taper :

```
?factorial
```

Question 1.3 Nous n'avons pas eu le temps de le discuter en cours cette année, mais la façon classique de juger de l'adéquation entre des données comme celles de la table et un modèle ajusté (les prédictions que vous venez de faire à la question précédente) est de calculer la statistique du χ^2:

$$Q = \sum_{i=0}^{12} \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i} \xrightarrow{L} \chi^2_{11},$$

où χ^2_{11} désigne une loi du χ^2 à 11 degrés de liberté et **où la convergence en loi a lieu quand le modèle est correcte**. Vous calculerez cette statistique et vous trouverez la probabilité pour qu'une variable aléatoire de loi χ^2_{11} **excède** la valeur trouvée. Que concluez vous ? Auriez-vous des suggestions à faire pour modifier le modèle (en particulier en vous aidant du graphe précédent) ?

Aide Je vous rappelle que les méthodes liées à la loi du χ^2 sont disponibles dans `SciPy` avec, par exemple :

```
from scipy.stats import chi2
```

Vous pouvez accéder à la documentation correspondante avec, par exemple :

```
?chi2
```

Dans les méthodes associées à la loi du χ^2, le paramètre formel `df` (un raccourci pour *degrees of freedom*) correspond au nombre de degrés de liberté de la loi considérée.

Question 1.4 Vous compenserez la carence théorique du cours où nous n'avons pas justifié la convergence en loi de la dernière équation en effectuant une simulation. Pour cela vous remarquerez que la table de données peut-être vue comme le résultat **d'un tirage multinomial** dans lequel 6115 « boules » (les familles) sont placées indépendemment les unes des autres dans 13 « boîtes » (le nombre de filles) différentes avec des probabilités : $p_0, p_1, \ldots, p_{12}$. Vous supposerez donc que votre $\hat{\theta}$ est la vraie valeur du paramètre et que votre modèle est bon. Vous simulerez alors 1000 tables de données — vous n'oublierez pas de spécifier la graine du générateur et chaque table contiendra un total de 6115 familles réparties suivant le même vecteur de probabilités — ; pour chaque table simulée, vous obtiendrez l'EMV, puis la statistique Q ci-dessus. Vous construirez alors un graphe de la fonction de répartition empirique de vos Q avec une bande de confiance à 99 % (vous utiliserez les fonctions `plot_fre` et `plot_fre_band` définies dans le cours) et vous y superposerez le graphe de la fonction de répartition d'un loi du χ^2_{11}.

Aide Le module `numpy.random` fournit la mise en œuvre d'un générateur de loi multinomiale :

```
from numpy.random import multinomial
```

1.1 Réponses

Le jeu de données est assez « célèbre » ; il a était publié par Geissler en 1889 pour une étude qui ne portait que sur la Saxe (et non sur l'ensemble de l'Allemagne).

Question 1.1 Il était en fait très simple d'obtenir la réponse en « prenant un peu de recul » — un examen, avec le stress qui lui est associé, ne présente pas franchement une situation propice à ce genre de réflexion et je ne m'attendais pas à ce que vous procédiez comme cela — ; en effet, si au sein de chaque famille les naissances sont considérées comme des événements indépendants les uns des autres (au sens statistique du terme !), pour ce qui est du sexe de l'enfant, avec une probabilité identique θ_i d'avoir une fille, alors le nombre de filles d'un couple ayant 12 enfants peut être vu comme le résultat de 12 épreuves de Bernoulli IID successives et l'espérance du nombre de filles devient : $12\times\theta_i$. Si on suppose en plus que la probabilité d'avoir une fille est la même (θ) pour chacune des 6115 familles et que le sexe des enfants nés est (statistiquement) indépendant d'une famille à l'autre, alors les 12×6115 naissances qui constituent l'échantillon peuvent être vues comme autant d'épreuves de Bernoulli IID avec une probabilité θ de succès (naissance d'une fille). L'espérance du nombre de filles dans l'échantillon devient donc : $12\times 6115\times\theta$ et l'estimateur du maximum de vraisemblance pour ce type de tirage n'est autre que le rapport empirique : nombre de filles / nombre de naissances.

Concrètement, après avoir rentré « à la main » les données dans `Python`:

```
nombre_de_filles = arange(13)
nombre_de_familles = array([7,45,181,478,829,1112,1343,1033,670,286,104,24,3])
```

Nous obtenons :

```
sum(nombre_de_filles*nombre_de_familles)/12./sum(nombre_de_familles)
```

```
0.48078495502861818
```

Maintenant, le type de réponse que j'attendais (même si ce n'est pas nécessairement le seul que j'accepte). Les données observées étaient nécessairement un n-uplet du type $(x_1, x_2, \dots, x_{6115})$ où chaque $x_i \in \{0, 1, \dots, 12\}$. Ce n-uplet est pour nous une observation d'un n-uplet de variables aléatoires $(\mathrm{X}_1, \mathrm{X}_2, \dots, \mathrm{X}_{6115})$ où, d'après les hypothèses explicitement formulées dans l'énoncer, les X_i sont IID de loi binomiale $\mathcal{B}(12, \theta)$ (12 tirages avec une probabilité de succès θ pour chacun d'eux). Nous pouvons donc immédiatement écrire :

$$\Pr\{\mathrm{X}_1 = x_1, \mathrm{X}_2 = x_2, \dots, \mathrm{X}_{6115} = x_{6115}\} = \prod_{i=1}^{6115} \frac{12!}{(12-x_i)!\,x_i!}\,\theta^{x_i}\,(1-\theta)^{12-x_i}\,.$$

Nous pouvons alors réarranger le produit du membre de droite en groupant les termes ayant la même valeur de x_i. Notons donc $n_j = \sum_{i=1}^{6115} \delta_{j,x_i}$, où $j = 0, 1, \dots, 12$ et où $\delta_{j,k}$ désigne le symbole de Kronecker, ce qui permet de réécrire notre probabilité conjointe :

$$\Pr\{\mathrm{X}_1 = x_1, \mathrm{X}_2 = x_2, \dots, \mathrm{X}_{6115} = x_{6115}\} = \prod_{j=0}^{12} \left(\frac{12!}{(12-j)!\,j!}\right)^{n_j} \left(\theta^j\,(1-\theta)^{12-j}\right)^{n_j}\,.$$

Remarquez que les n_j que nous venons d'introduire sont les nombres constituant la seconde ligne de notre table de données. La log-vraisemblance de θ devient donc :

$$l(\theta) = \mathrm{K} + \sum_{j=0}^{12} n_j \left(j \log(\theta) + (12 - j) \log(1 - \theta) \right) ,$$

où

$$\mathrm{K} = \log \left(\prod_{j=0}^{12} \left(\frac{12!}{(12-j)!\, j!} \right)^{n_j} \right)$$

ne dépend pas du paramètre θ. Maintenant, le fait que, dans la table de données, tous les $n_j \neq 0$, nous permet d'affirmer que ni 0 ni 1 ne maximisent notre (log-)vraisemblance et donc que l'EMV $\hat{\theta}$ est un point intérieur de $[0, 1]$ (considération mathématiques mises à part, une valeur 0 ou 1 pour θ entraînerait une fin assez rapide de l'espèce humaine !). Nous pouvons donc sans arrière pensée chercher $\hat{\theta}$ comme solution de l'équation de vraisemblance : $d \log(\hat{\theta})/d\theta = 0$. Nous devons donc avoir :

$$\sum_{j=0}^{12} \frac{j n_j}{\hat{\theta}} = \sum_{j=0}^{12} \frac{12 n_j - j n_j}{1 - \hat{\theta}} ,$$

là une petite manipulation algébrique nous donne immédiatement :

$$\hat{\theta} = n_{\mathrm{F}} / n_{\mathrm{T}} ,$$

où $n_{\mathrm{T}} = 12 \sum_{j=0}^{12} n_j$ est le nombre total de naissances et $n_{\mathrm{F}} = \sum_{j=0}^{12} j n_j$ est le nombre de petites filles. D'où notre EMV :

```
theta_chapeau = sum(nombre_de_familles*nombre_de_filles)/12./sum(nombre_de_familles)
```

En toute rigueur, nous devons vérifier que la dérivée seconde de la log-vraisemblance est négative en $\hat{\theta}$. Ayant introduit n_{F} et n_{T}, nous pouvons réécrire notre log-vraisemblance de la façon suivante :

$$l(\theta) = \mathrm{K} + n_{\mathrm{F}} \left(\log(\theta) - \log(1 - \theta) \right) + n_{\mathrm{T}} \log(1 - \theta) ,$$

d'où

$$\frac{dl(\theta)}{d\theta} = n_{\mathrm{F}} \left(\frac{1}{\theta} + \frac{1}{1 - \theta} \right) - \frac{n_{\mathrm{T}}}{1 - \theta}$$

et

$$\frac{d^2 l(\theta)}{d\theta^2} = -\frac{n_{\mathrm{F}}}{\theta^2} - \frac{n_{\mathrm{T}} - n_{\mathrm{F}}}{(1 - \theta)^2} < 0 \quad \text{pour} \quad \theta \in [0, 1] .$$

Question 1.2 Nous définissons une fonction qui renvoie un vecteur de probabilités dont chaque élément donne la probabilité d'avoir un nombre de filles correspondant à son indice (rappelez vous que les indices commencent à 0 dans `Python`) :

```
from scipy.misc import factorial
def proba_v(theta):
    return array([factorial(12)/factorial(12-i)/factorial(i)*theta**i*(1-theta)**(12-i)
                  for i in range(13)])
```

Nous en déduisons notre vecteur de prédictions :

```
predictions_v = proba_v(theta_chapeau)*sum(nombre_de_familles)
```

D'où le graphe (figure 1) :

```
plot(arange(13),nombre_de_familles-predictions_v,'ro')
grid()
xlabel("Nombre de filles")
ylabel(u"Différence entre observations et prédictions")
```

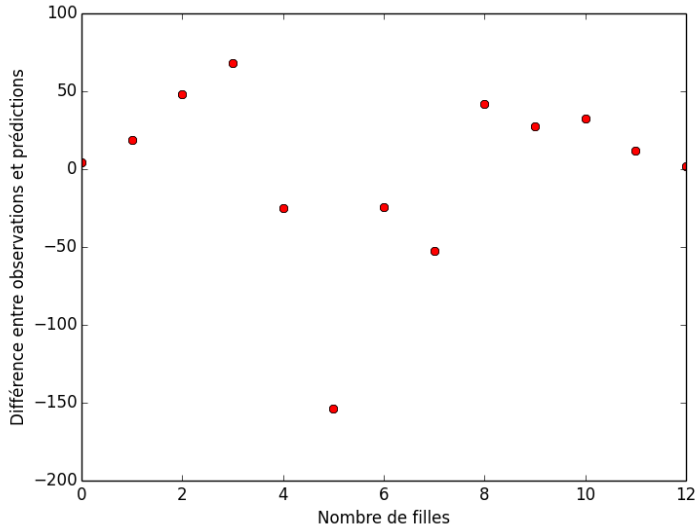

FIG.1: Différence entre observations et prédictions pour le nombre de filles dans des familles de 12 enfants. Prédictions basées sur une probabilité uniforme d'avoir une fille pour chacune des 6115 × 12 naissances considérées dans l'étude. Les naissances sont supposées statistiquement indépendantes et la probabilité d'avoir une fille est estimée en maximisant la vraisemblance.

On constate un excès aux extrêmes et un déficit au centre. Dit autrement on a trop de familles avec entre 0 et 3 filles d'une part et entre 8 et 12 d'autre part ; on en a trop peu avec entre 4 et 7 filles. À ce stade, nous n'avons pas de « mètre étalon » nous permettant de jauger l'importance à accorder à ces déviations, mais leur caractères systématique suggère que notre modèle n'a pas capturé de façon satisfaisante le mécanisme à l'œuvre dans ces données. La question suivante à pour but de quantifier cette impression.

Question 1.3 Pour Q nous obtenons :

```
Q_chapeau = sum((nombre_de_familles-predictions_v)**2/predictions_v)
round(Q_chapeau,1)
```

```
110.5
```

D'où la probabilité demandée :

```
from scipy.stats import chi2
1-chi2.cdf(Q_chapeau,11)
```

```
0.0
```

Nous avons donc une probabilité essentiellement nulle pour que notre modèle, s'il est correct, donne une observation aussi large que celle que nous avons observée pour Q. En langage de statisticien, nous pouvons rejeter le modèle. La façon classique d'aller plus loin, à partir de là est de modifier le modèle en amendant l'hypothèse du même θ_i pour chaque famille. Cela revient à construire un modèle dans lequel certaines famille ont un $\theta_i > \hat{\theta}$, alors que d'autre ont un $\theta_i < \hat{\theta}$. Si cela vous intéresse vous pouvez consulter l'article de Lindsey et Altham (1998)[1].

Question 1.4 On procède comme suit :

```
from numpy.random import multinomial
Q_bootstrap = zeros(1000) ## vecteur de résultats
total_familles = sum(nombre_de_familles)
p_v = proba_v(theta_chapeau) ## vecteur de proba pour les simulations
seed(20110928) ## NE PAS OUBLIER LA GRAINE !
## La variable suivante contient une matrice avec 1000 lignes (notre
## nombre de table simulées) et 12 colonnes. Chaque ligne est une
## table simulée.
les_tables = multinomial(total_familles,p_v,size=len(Q_bootstrap))
for r_idx in range(len(Q_bootstrap)):
    ma_table = les_tables[r_idx,:] ## extraction de r_idx-ième table simulée
    ## la variable suivante contient l'EMV calculé sur les données simulées
    theta_sim = sum(ma_table*nombre_de_filles)/12./total_familles
    ## ensuite, le vecteur de prédictions
    predictions_sim = proba_v(theta_sim)*total_familles
    ## puis le calcul de Q et sa sauvegarde dans le vecteur de résultats.
    Q_bootstrap[r_idx] = sum((ma_table-predictions_sim)**2/predictions_sim)
```

Pour contruire le graphe demandé, nous commençons par évaluer la définition de `plot_fre_band` (copie du code du cours) :

```
def plot_fre_band(x, alpha = 0.95, domain = None, color='black', lw=1):
    X = sort(x)
    n = len(x)
    Y = arange(1.,n+1)/n
    epsilon = sqrt(log(2/(1-alpha))/2/n)
    U = clip(Y+epsilon,0,1)
    L = clip(Y-epsilon,0,1)
    if domain is None:
        domain = [floor(X[0]),ceil(X[n-1])]
    for i in range(n-1):
        plot(X[i:(i+2)],[U[i],U[i]],color=color,lw=lw)
        plot(X[i:(i+2)],[L[i],L[i]],color=color,lw=lw)
    xlim(domain)
    ylim([0,1])
    hlines(0,domain[0],X[0],colors=color,lw=lw)
    hlines(1,X[n-1],domain[1],colors=color,lw=lw)
```

On obtient la figure 2 avec :

```
plot_fre_band(Q_bootstrap,0.99)
xlabel("Statistique Q")
ylabel(u"Fréquence")
xx = linspace(0,35,201)
plot(xx,chi2.cdf(xx,11),color='red')
```

[1] Lindsey et Altham (1998) Analysis of the Human Sex Ratio by Using Overdispersion Models. *Applied Statistics* **47**, 149-157.

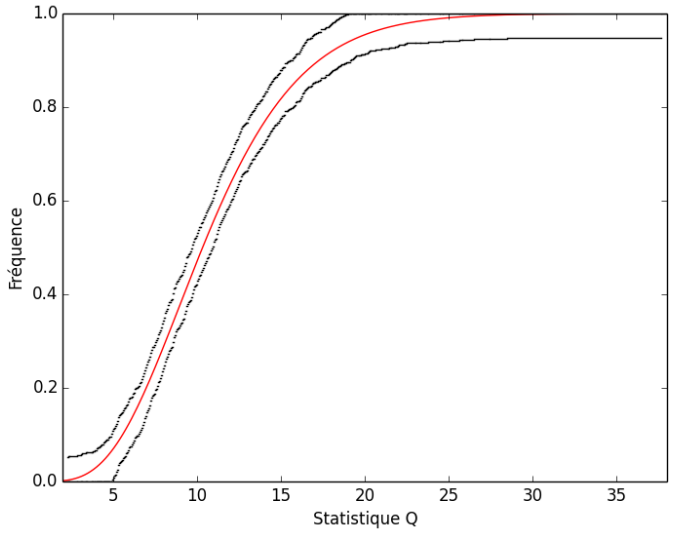

FIG.2: En rouge, la fonction de répartition d'une loi du χ^2 à 11 degrés de liberté. En noir, une bande de confiance à 99 % calculer à partir de 1000 répliques de la statistique Q simulées avec le modèle ajusté des questions 1.1 et 1.2.

2 Problème 2 : construction d'un histogramme (5 points)

Vous téléchargerez les données du Sloan Digital Sky Survey[2] depuis le site de Larry Wasserman[3] et vous construirez un histogramme en choisisant le nombre de classes de ce dernier en minimisant l'erreur quadratique intégrée moyenne.

Aide Les données peuvent être chargées dans `Python` avec (c'est un peu compliqué) :

```
from urllib2 import urlopen
u = urlopen('http://www.stat.cmu.edu/~larry/all-of-nonpar/=data/galaxy.dat')
sdss = array([float(x.lstrip().rstrip('\n').split(' ')[-1])
              for x in u.readlines()])
u.close()
```

Vous avez un long exemple de construction d'histogramme au début du cours et trois en fin de cours (dans l'exercice sur la structure de l'ADN). L'histogramme que vous obtiendrez ne sera pas nécessairement « joli » (il pourra même avoir un aspect de « montagnes russes »).

2.1 Réponse

Nous chargeons les données comme indiqué dans l'énoncer puis nous évaluons ensuite la définition de `fait_fre` (copie du cours) :

[2] http://www.stat.cmu.edu/~larry/all-of-nonpar/=data/galaxy.dat
[3] http://www.stat.cmu.edu/~larry/all-of-nonpar/data.html

```
def fait_fre(echantillon):
    y = sort(echantillon)
    n = float(len(echantillon))
    def fre(x):
        return sum(y <= x)/n
    return fre
```

Puis la définition de `j_chapeau` (copie du cours) :

```
def j_chapeau(echantillon,m):
    fre = fait_fre(echantillon)
    n = float(len(echantillon))
    x_min = min(echantillon)
    x_max = max(echantillon)
    limites_classes = linspace(x_min,x_max,m+1)
    p_chapeau = array([fre(limites_classes[i]) for i in range(m+1)])
    p_chapeau = diff(p_chapeau)
    return (2 - (n+1)*sum(p_chapeau**2))*m/(n-1)
```

Enfin, les définitions de `histogramme` et `histogramme_fct` (copie du cours) :

```
def histogramme(echantillon,m):
    fre = fait_fre(echantillon)
    n = float(len(echantillon))
    x_min = min(echantillon)
    x_max = max(echantillon)
    limites_classes = linspace(x_min,x_max,m+1)
    p_chapeau = array([fre(limites_classes[i]) for i in range(m+1)])
    f_chapeau = diff(p_chapeau)/diff(limites_classes)
    return (limites_classes, f_chapeau)

def histogramme_fct(echantillon,m):
    fre = fait_fre(echantillon)
    x_min = min(echantillon)
    x_max = max(echantillon)
    limites_classes = linspace(x_min,x_max,m+1)
    p_chapeau = array([fre(limites_classes[i]) for i in range(m+1)])
    f_chapeau = diff(p_chapeau)/diff(limites_classes)
    def histo(x):
        if x < x_min or x > x_max:
            return 0
        else:
            return f_chapeau[max(where(x >= limites_classes)[0])]
    return histo
```

On applique alors les fonctions comme dans le cours (il suffit d'adapter ce qui a été fait dans l'exercice sur la structure de l'ADN) :

```
mm_sdss = arange(1,len(sdss))
j_chapeau_sdss = array([j_chapeau(sdss,i) for i in mm_sdss])
mm_sdss[argmin(j_chapeau_sdss)]
```

```
597
```

Et on obtient l'histogramme demandé (figure 3) :

```
cb_sdss,f_chapeau_sdss = histogramme(sdss,mm_sdss[argmin(j_chapeau_sdss)])
subplot(121)
plot(mm_sdss,j_chapeau_sdss,color="black",lw=2)
grid()
xlabel("Nombre de classes")
ylabel(u"Erreur quadratique intégrée moyenne")
subplot(122)
plot(cb_sdss[1:],f_chapeau_sdss,ls='steps',color='black',lw=2)
ylabel(u'Densité estimée')
xlabel("Redshift")
```

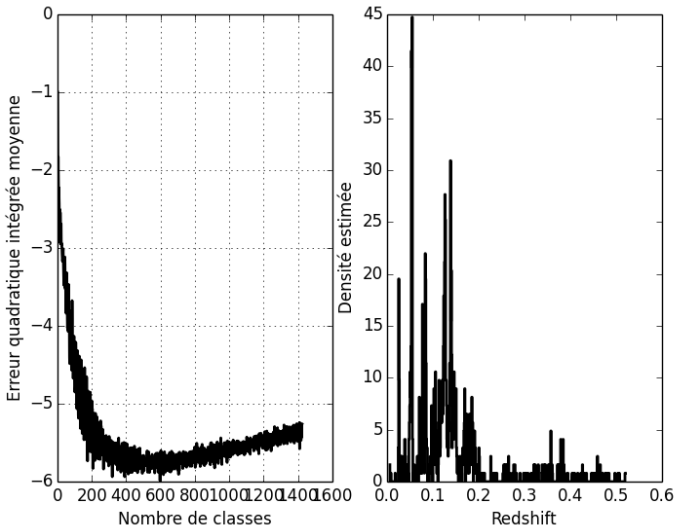

FIG.3: À gauche, l'erreur quadratique intégrée moyenne en fonction du nombre de classes de l'histogramme. À droite, l'estimation de densité construite avec le nombre optimal de classe.

Remarque Si vous regardez le bouquin de Wasserman (*All of Nonparametric Statistics*), vous verrez qu'il a d'abord réduit l'échantillon en ne gardant que les observations pour lesquelles le *redshift* est < 0,2.

3 Problème 3 : un petit problème de génétique (8 points)

On considère une population d'individus diploïdes[4] à reproduction sexuée. On s'intéresse de plus au gène codant l'haptoglobine (une protéine du plasma sanguin) présent sous deux versions différentes (allèles) A et *a* dans notre population. Chaque individu de la population présente alors l'un des trois phénotypes suivants :

- *Hp1-1* s'il est AA (c.-à-d. qu'il a deux versions identiques, A et A, du gène) ;

[4]Chaque chromosome est présent par paire (comme chez nous les humains) dans toute cellule non reproductrice (ou toute cellule qui n'est pas une gamète). Donc chaque cellule (non reproductrice) contient **deux copies de chaque gène**.

- *Hp1-2* s'il est Aa ;
- *Hp2-2* s'il est aa.

Un modèle assez simple à dériver prédit que si la population est à l'équilibre[5] et que si on désigne par θ la fréquence de l'allèle a dans la population alors :

$$\Pr(\mathrm{AA}) = (1-\theta)^2, \quad \Pr(\mathrm{A}a) = 2\theta(1-\theta), \quad \Pr(aa) = \theta^2 .$$

Nous disposons d'un échantillon de 190 individus (d'une population que nous supposerons à l'équilibre) dont les phénotypes sont :

Phénotype	*Hp1-1*	*Hp1-2*	*Hp2-2*
Nombre d'individus	10	68	112

Question 3.1 Vous trouverez l'EMV (estimateur du maximum de vraisemblance) de θ.

Question 3.2 Vous trouverez la variance asymptotique de l'EMV (vous supposerez que les « conditions de régularité » sont vérifiées ; l'inverse de la dérivée seconde de la log vraisemblance évalué à l'EMV devrait alors vous fournir la réponse).

Question 3.3 Vous obtiendrez un intervalle de confiance à 95 % avec la méthode de Wald.

Question 3.4 Vous construirez un graphe de la log vraisemblance puis vous y ajouterez l'approximation quadratique de cette dernière.

Question 3.5 Vous obtiendrez un intervalle de confiance à 95 % avec la méthode du rapport de vraisemblance. La fonction `brentq` de `scipy.optimize` pourra être utile à ce stade.

Question 3.6 Si vous avez le temps, vous étudierez, par simulations, les probabilités de recouvrements empiriques des deux intervalles, comme nous l'avons fait dans le cours (en particulier à la fin de l'exercice sur la structure de l'ADN).

3.1 Réponses :

On a ici, sous une forme à peine différente, le même problème que dans la première question, au sens où c'est encore un modèle multinomiale qui est employé. On a 3 catégories (au lieu de 13) et le modèle du vecteur de probabilités est différent.

Question 3.1 En notant $(n_{\mathrm{AA}}, n_{\mathrm{A}a}, n_{aa})$ le vecteur des observations (la seconde la table des données), on obtient immédiatement l'expression suivante pour la log-vraisemblance :

$$l(\theta) = \mathrm{K} + (2n_{\mathrm{AA}} + n_{\mathrm{A}a}) \log(1-\theta) + (n_{\mathrm{A}a} + 2n_{aa}) \log\theta ,$$

d'où

$$\frac{dl(\theta)}{d\theta} = -\frac{2n_{\mathrm{AA}} + n_{\mathrm{A}a}}{1-\theta} + \frac{n_{\mathrm{A}a} + 2n_{aa}}{\theta}$$

et l'EMV est :

$$\hat{\theta} = \frac{n_{\mathrm{A}a} + 2n_{aa}}{2n_{\mathrm{T}}} ,$$

[5] Les fractions d'individus dans chaque catégorie ne changent pas d'une génération à l'autre.

où $n_{\mathrm{T}} = n_{\mathrm{AA}} + n_{\mathrm{A}a} + n_{aa}$. l'EMV ainsi définit est bien un maximum puisque :

$$\frac{d^2 l(\theta)}{d\theta^2} = -\frac{2n_{\mathrm{AA}} + n_{\mathrm{A}a}}{(1-\theta)^2} - \frac{n_{\mathrm{A}a} + 2n_{aa}}{\theta^2} < 0 \quad \text{pour} \quad \theta \in [0,1]\,.$$

La valeur numérique de l'EMV est donc :

```
Hp = array([10,68,112])
total_Hp = sum(Hp)
theta_chapeau = (Hp[1]+2*Hp[2])/2./total_Hp
theta_chapeau
```

```
0.76842105263157889
```

Question 3.2 En combinant nos expressions de l'EMV et de la dérivée seconde de la log-vraisemblance (remarquez que $n_{\mathrm{A}a} + 2n_{aa} = 2n_{\mathrm{T}}\hat{\theta}$ et que $n_{\mathrm{A}a} + 2n_{\mathrm{AA}} = 2n_{\mathrm{T}}(1-\hat{\theta})$) il vient :

$$\frac{d^2 l(\theta)}{d\theta^2} = -2n_{\mathrm{T}}\left(\frac{1-\hat{\theta}}{(1-\theta)^2} + \frac{\hat{\theta}}{\theta^2}\right),$$

d'où

$$\frac{d^2 l(\hat{\theta})}{d\theta^2} = -\frac{2n_{\mathrm{T}}}{\hat{\theta}(1-\hat{\theta})}$$

et (comme spécifié dans l'énoncer)

$$\widehat{es}^2 = \frac{\hat{\theta}(1-\hat{\theta})}{2n_{\mathrm{T}}}$$

La variance asymptotique a donc ici la valeur :

```
es2_chapeau = theta_chapeau*(1-theta_chapeau)/2/total_Hp
es2_chapeau
```

```
0.00046828983816882931
```

Question 3.3 Application directe du cours, l'intervalle de confiance à 95 % avec la méthode de Wald :

```
[theta_chapeau - 1.96*sqrt(es2_chapeau),theta_chapeau + 1.96*sqrt(es2_chapeau)]
```

```
[0.72600664184574604, 0.81083546341741175]
```

Question 3.4 En suivant l'exemple de l'exercice sur la structure de l'ADN, on construit le graphe demandé de la façon suivante (figure 4) :

```
tt = linspace(theta_chapeau-3*sqrt(es2_chapeau),theta_chapeau+3*sqrt(es2_chapeau),201)
plot(tt,(2*Hp[0]+Hp[1])*log(1-tt)+(Hp[1]+2*Hp[2])*log(tt),color='black',lw=2)
xlabel("$\\theta$")
ylabel("Log-vraisemblance")
plot(tt,(2*Hp[0]+Hp[1])*log(1-theta_chapeau)+
     (Hp[1]+2*Hp[2])*log(theta_chapeau)-(tt-theta_chapeau)**2/2./es2_chapeau,
     color='red',lw=2)
```

Question 3.5 En suivant toujours l'exemple de l'exercice sur la structure de l'ADN, on obtient l'intervalle de confiance à 95 % avec la méthode du rapport de vraisemblance :

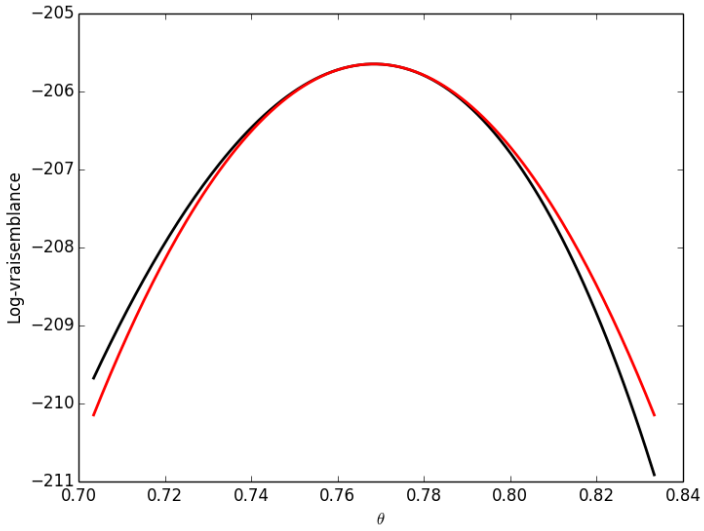

FIG.4: La log-vraisemblance (noir) et son approximation quadratique (rouge).

```
from scipy.optimize import brentq
def log_vrai_Hp(theta): return (2*Hp[0]+Hp[1])*log(1-theta)+(Hp[1]+2*Hp[2])*log(theta)
k_0p05 = chi2.ppf(1-0.05,1)
def cible_Hp(theta): return log_vrai_Hp(theta_chapeau) - log_vrai_Hp(theta) - k_0p05/2.
[brentq(cible_Hp,max(theta_chapeau-6*sqrt(es2_chapeau),0.1),theta_chapeau),
 brentq(cible_Hp,theta_chapeau,min(theta_chapeau+6*sqrt(es2_chapeau),0.9))]
```

```
[0.724291931713986, 0.8089437882373592]
```

Question 3.6 Il suffit de continuer avec l'exemple de l'exercice sur la structure de l'ADN…

```
wald = []
rv = []
n_rep = 1000
p_hapto_v = array([(1-theta_chapeau)**2,
                   2*theta_chapeau*(1-theta_chapeau),
                   theta_chapeau**2])
seed(20061001)
hapto_tables = multinomial(total_Hp,p_hapto_v,size=n_rep)
for r_idx in range(n_rep):
    Hp_sim = hapto_tables[r_idx,:]
    theta_sim = (Hp_sim[1]+2*Hp_sim[2])/2./total_Hp
    es_sim = sqrt(theta_sim*(1-theta_sim)/2./total_Hp)
    wald.append([theta_sim-1.96*es_sim,theta_sim+1.96*es_sim])
    def log_vrai_Hp(theta): return (2*Hp_sim[0]+Hp_sim[1])*log(1-theta)+\
        (Hp_sim[1]+2*Hp_sim[2])*log(theta)
    def cible_Hp(theta): return log_vrai_Hp(theta_sim) - log_vrai_Hp(theta) - k_0p05/2.
    rv.append([brentq(cible_Hp,max(theta_sim-6*es_sim,0.1),theta_sim),
               brentq(cible_Hp,theta_sim,min(theta_sim+6*es_sim,0.9))])
```

Ce qui nous donne pour nos probabilités de recouvrement empiriques pour la méthode de wald :

```
sum([x[0] <= theta_chapeau <= x[1] for x in wald]) / 1000.
```

```
0.94499999999999995
```

et pour le rapport de vraisemblance :

```
sum([x[0] <= theta_chapeau <= x[1] for x in rv]) / 1000.
```

```
0.94199999999999995
```

Vous pourrez facilement vérifier en utilisant la fonction de répartition d'une loi $\mathcal{B}(1000, 0.95)$ que ces deux valeurs sont compatibles avec une probabilité de recouvrement de 0,95.

4 Remarques générales

Dans le fichier « texte » — c'est-à-dire au format `ASCII` — que vous m'enverrez à la fin de l'examen, je supposerez que vous avez utilisé la version de `IPython` installée sur vos machines et que vous avez exécuté la commande magique `%pylab` en début de session. Vous pouvez décider d'utiliser une autre interface `Python` (ou même d'utiliser `R`) mais alors je vous demande de le préciser par un commentaire en début de fichier ; je vous demande aussi d'écrire alors l'ensemble des commandes me permettant de reproduire vos résultats.

Comme précisé dans le cours, n'oubliez pas de fixer explicitement dans le fichier que vous m'enverrez, la graine du générateur de nombres aléatoires que vous avez employée (idéalement avant chaque simulation, pour que je puisse reproduire exactement chacune d'elles).

Pensez à mettre des labels sur les axes de vos figures.