UNIVERSITÉ DE MONTPELLIER

Les Modèles Linéaires Mixtes

HMMA 307 : Modèles Linéaires Avancés

Walid KANDOUCI

Table des matières

1 Introduction **2**

2 Modèles linéaires mixtes **2**

2.1 Définition 2

3 Application **3**

3.1 Ordinary Least Squares (OLS) et Generalised Linear mixed Models (GLM) 5

3.2 LMM 6

4 Conclusion **7**

1 Introduction

On appelle modèles mixtes des modèles comportant à la fois des facteurs à effets fixes (ces effets entrant dans la définition de la moyenne du modèle) et des facteurs à effets aléatoires (ces effets entrant, quant à eux, dans la définition de la variance du modèle).

Nous allons voir dans ce projet une définition plus détaillé sur les modèles linéaire mixte (appelé aussi LMM), ainsi qu'une application en utilisant le logiciel Python.

2 Modèles linéaires mixtes

La statistique traditionnelle est basé sur plusieurs hypothèses sur le maximum de vraisemblance et sur la distribution d'une loi normal. Dans un exemple de régression multiple, ces hypothèses peuvent ne pas être respecté dans le cas de non-indépendance de nos données.

Sachant que nos données son des matrices de dimension $(n \times p)$ ou p représente le nombre de variables et n le nombre d'observations, il peux y avoir deux types de données dans nos données :

- Corrélation des variables
- Corrélation des observations

Dans les deux cas, l'inverse de la matrice de nécessaire à la solution du modèle linéaire est singulière, car son déterminant est proche de zéro en raison de variables ou d'observations corrélées.

Ce problème ce manifeste particulièrement quand on travail sur des données a grande dimension $(p >> n)$ quand les variables peuvent devenir redondantes et corrélées :

Pour fixer ce problème, on peut par exemple sélectionner des variables informatives à l'aide de l'algorithme de LASSO, ou de Ridge et en prenant en compte de la corrélations des observation à l'aide de la modélisation aléatoire dans le modèle linéaire mixte.

2.1 Définition

Les modèles mixtes une extension du modèle linéaire général qui prend en compte la structure hiérarchique des données. Ils conviennent à de nombreux types de données telles que les données groupées, les données à mesures répétées, les données longitudinales, ainsi que les combinaisons de ces trois.

On rappelle la formule du modèle de régression général :

$$y = X\beta + \epsilon$$

ou :

- X est notre matrice $(N \times p)$
- β est un vecteur colonne $p \times 1$ qui contient les coefficients de régression pour chaque variable indépendante du modèle
- $epsilon$ est un vecteur colonne $1 \times p$ vecteur qui contient les erreurs (résidus) du modèle

Le modèle à effets mixtes est une extension et modélise les effets aléatoires d'une variable de regroupement :

$$y = X\beta + Zu + \epsilon$$

Ou :

- Z est une matrice $(N \times p)$ qui contient les valeurs observées pour chaque individu N pour chaque covariable q des effets aléatoires.
- β est un vecteur colonne $p \times 1$ u est un vecteur $(q \times 1)$ qui contient les effets aléatoires des q covariables de Z

3 Application

Les modèles linéraires mixtes doivent être utilisé lorsqu'il y a une sorte de regroupement entre les observations.

Ici nous allons utilisé les données **"sleepstudy"** qui sont des données que l'on peut trouver dans le package **"lme4"** que l'on va exporter afin de faire notre étude avec python.

Ce jeu de données représente une étude sur le temps de réaction sur la privation de sommeil :

- Au jour 0, les sujets ont dormi normalement
- À partir de cette nuit-là, ils étaient limités à 3 heures de sommeil par nuit
- Les observations représentent le temps de réaction moyen sur une série de tests donnés chaque jour à chaque sujet

Ces données proviennent de l'étude décrite dans *Belenky et al.(2003)* pour le groupe privé de sommeil et pendant les 10 premiers jours de l'étude, jusqu'à la période de récupération.

```
data = pd.read_csv("C:\\Users\Walid\Documents\sleepstudy.csv")
data.index = data[data.columns[0]]
data = data[data.columns[1:4]]
```

	Reaction	Days	Subject
1	249.5600	0	308
2	258.7047	1	308
3	250.8006	2	308
4	321.4398	3	308
5	356.8519	4	308

Notre jeu de données est constitué de 180 individus ainsi que de 3 variables :

- **Reaction :** Temps de réaction en moyenne (en milisecondes)
- **Days :** Nombre de jours de privation de sommeil
- **Subject :** Numéro du sujet sur lequel l'observation a été faite.

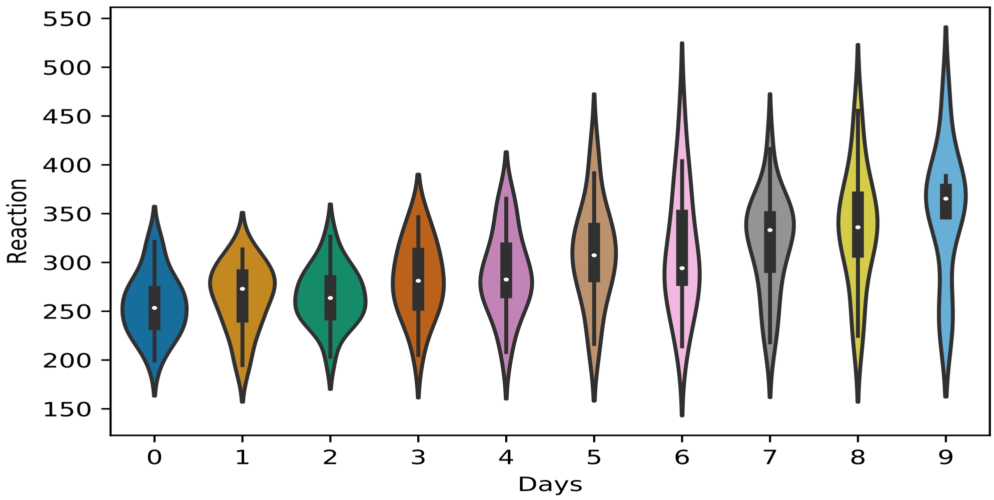
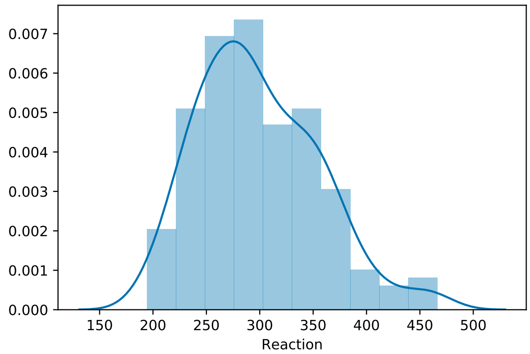


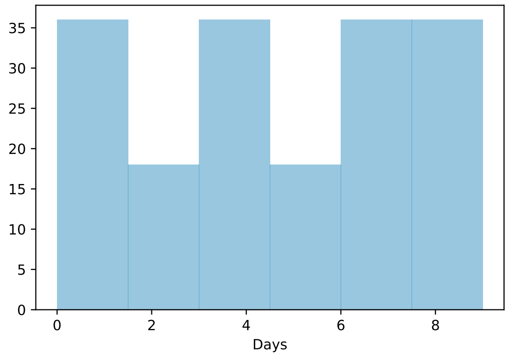
FIGURE 1 – violins-plots de nos données, jours /réactions.

On peut voir ici que plus en avance dans le temps, plus nous avons des variations en matière de temps de réactions.

Regardons maintenant nos variables jours et temps de réactions :



(a) Distribution des réactions



(b) Distribution des jours

FIGURE 2 – Analyse de nos variables "Reaction" et "Days"

On remarque un temps de réactions élevé dans l'intervalle $[250, 350]$ ainsi qu'un taux de privation de sommeil élevé dans les 4 derniers jours l'étude.

On regarde maintenant si le temps de réactions dépends des jours qui passent :

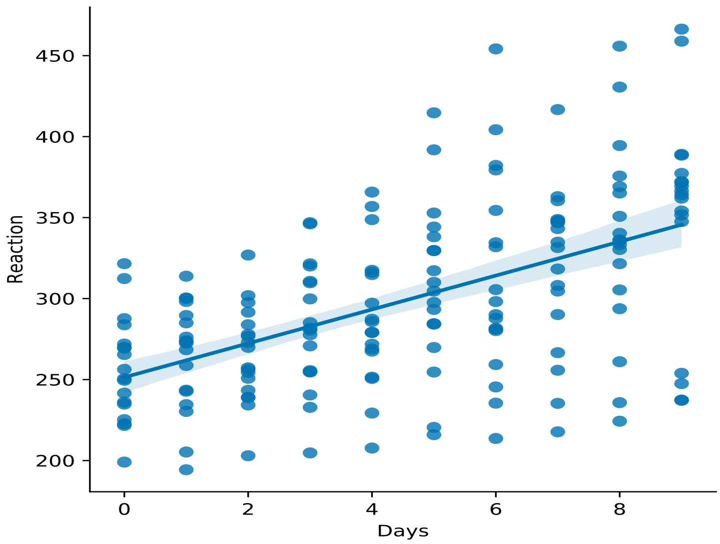


FIGURE 3 – Représentation du temps de réaction en fonctions des jours

On remarque que le temps réaction par rapport aux jours a une tendance à la hausse avec des variations entre les jours et les individus (il existe une relation croissante entre le temps de réaction et les jours)

3.1 Ordinary Least Squares (OLS) et Generalised Linear mixed Models (GLM)

On ajuster les modéles OLS et GLM pour l'effet des jours pour chaque individu.
Le modéles OLS s'écrit :

$$\mathrm{Y} = \beta_0 + \Sigma_{\mathrm{j}=1..p}\beta_{\mathrm{j}} X_{\mathrm{j}} + \varepsilon$$

ou :

- Y désigne la variable dépendante
- β_0 constante du modèle
- X_j désigne la $\mathrm{j}^{\text{ème}}$ variable explicative du modèle ($j = 1$ a p),

```
# OLS
modelOLS = smf.ols("Reaction ~ Days", data, groups=data["Subject"])
resultOLS = modelOLS.fit()
print(resultOLS.summary())
```

Pour le modéles OLS, on obtient les résultats suivants :

OLS Regression Results

Dep. Variable :	Reaction	R-squared :	0.286
Model :	OLS	Adj. R-squared :	0.282
Method :	Least Squares	F-statistic :	71.46
Date :	Sat, 07 Nov 2020	Prob (F-statistic) :	9.89e-15
Time :	22 :09 :28	Log-Likelihood :	-950.15
No. Observations :	180	AIC :	1904.
Df Residuals :	178	BIC :	1911.
Df Model :	1		
Covariance Type :	nonrobust		

	Coef.	Std.Err.	z	$P > \|z\|$	[0.0250.975]
Intercept	251.4051	6.610	38.033	0.000	238.361 264.449
Days	10.4673	1.238	8.454	0.000	8.024 12.911

Omnibus :	1.455	Durbin-Watson :	0.695
Prob(Omnibus) :	0.483	Jarque-Bera (JB) :	1.128
Skew :	0.015	Prob(JB) :	0.569
Kurtosis :	3.387	Cond. No.	10.2

```
# GLM
modelGLM = smf.glm("Reaction ~ Days", data, groups=data["Subject"])
resultGLM = modelGLM.fit()
print(resultGLM.summary())
```

Pour les modéles GLM, on obtient les résultats suivants :

Generalized Linear Model Regression Results

Dep. Variable :	Reaction	No.Observations :	180
Model :	GLM	Df Residuals :	178
Model Family :	Gaussian	Df Model :	1
Link Function :	identity	Scale :	2276.7
Method :	IRLS	Log-Likelihood :	-950.15
Date :	Sat, 07 Nov 2020	Deviance :	4.0525e+05
Time :	22 :13 :58	Pearson chi2 :	4.05e+05
No. Iterations :	3		
Covariance Type :	nonrobust		

	Coef.	Std.Err.	z	$P > \|z\|$	[0.0250.975]
Intercept	251.4051	6.610	38.033	0.000	238.449 264.361
Days	10.4673	1.238	8.454	0.000	8.040 12.894

3.2 LMM

Maintenant, pour ajuster un modèle d'interception aléatoire, rappelez-vous que ce type de modèle permet à différents clusters (un groupe) d'avoir des interceptions différentes

```
#LMM
md = smf.mixedlm("Reaction ~ Days", data, groups=data["Subject"])
mdf = md.fit()
print(mdf.summary())
```

On obtient les résultats suivants :

Mixed Linear Model Regression Results

Model :	MixedLM	Dependent Variable :	Reaction
No. Observations :	180	Method :	REML
No. Groups :	18	Scale :	960.4568
Min. group size :	10	Log-Likelihood :	-893.2325
Max. group size :	10	Converged :	Yes
Mean group size :	10.0		

	Coef.	Std.Err.	z	$P > \|z\|$	[0.0250.975]
Intercept	251.405	9.747	25.794	0.000	232.302 270.508
Days	10.467	0.804	13.015	0.000	8.891 12.044
Group Var	1378.176	17.156			

On remarque que :

- Le nombre de groupes = 18
- Corrélation des observations

En calculant l'erreur quadratique moyenne, on obtient les résultats suivants :

```
y = data.Reaction
y_predict_LMM = resultLMM.fittedvalues
RMSE_LMM = sqrt(((y-y_predict_LMM)**2).values.mean())
results = pd.DataFrame()
results["Method"] = ["LMM"]
results["RMSE"] = RMSE_LMM
y_predict_GLM = resultGLM.fittedvalues
RMSE_GLM = sqrt(((y-y_predict_GLM)**2).values.mean())
results.loc[1] = ["GLM",RMSE_GLM]
y_predict_OLS = resultOLS.fittedvalues
RMSE_OLS = sqrt(((y-y_predict_OLS)**2).values.mean())
results.loc[2] = ["OLS",RMSE_OLS]
results
```

Method	RMSE
LMM	29.410624
GLM	47.448898
OLS	47.448898

On remarque que l'erreur quadratique moyenne du LMM est la plus petite, ce qui montre que cette méthode fourni un meilleur ajustement que les deux autres méthodes.

```
performance = pd.DataFrame()
performance["residuals"] = resultLMM.resid.values
performance["Days"] = data.Days
performance["predicted"] = resultLMM.fittedvalues
sns.lmplot(x = "predicted", y = "residuals", data = performance)
plt.savefig("figure6.pdf")
```

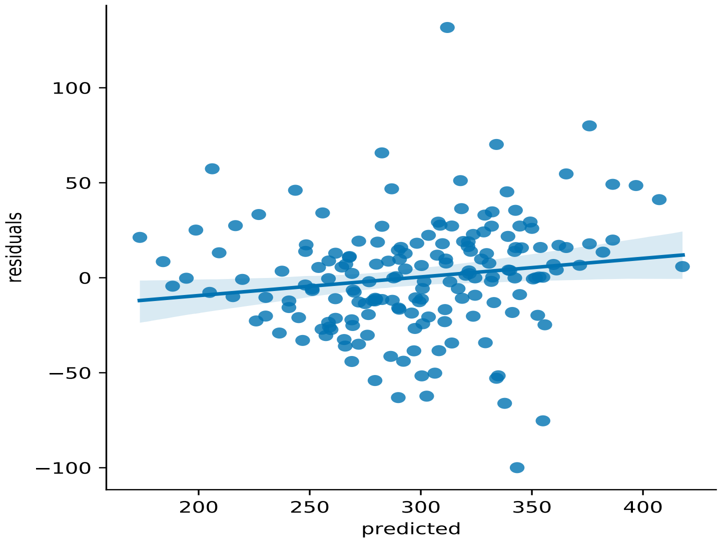


FIGURE 4 – Représentation du temps de réactions en fonctions des jours après ajustement LMM

Grâce a cette méthode, les résidus sont bien meilleurs qu'avant (le temps de réactions étant mieux répartis en fonction des jours)

4 Conclusion

Nous avons appris que le modèle linéaire mixte est utilisé pour tenir compte de la non-indépendance et donc de la non-normalité des données.

le modèle fourni généralement un meilleur ajustement et expliquent plus de variation dans les données comparé aux modéles OLS ou GLM.

Références

[1] How Linear Mixed Model Works (Nikolay Oskolkov)
https ://towardsdatascience.com/how-linear-mixed-model-works-350950a82911

[2] How Linear Mixed Model Works (Nikolay Oskolkov)
https ://www.statsmodels.org/stable/mixed_linear.html

[3] mixed_models (OJ Watson)
https ://www.kaggle.com/ojwatson/mixed-models

Code Python

```
%matplotlib inline

import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.tools.sm_exceptions import ConvergenceWarning
import matplotlib.pyplot as plt
from math import sqrt
import seaborn as sns
sns.set_palette("colorblind")

data = pd.read_csv("C:\\Users\Walid\Documents\sleepstudy.csv")
data.index = data[data.columns[0]]
data = data[data.columns[1:4]]

data.head(5)

sns.violinplot(x="Days", y='Reaction', data=data)
plt.savefig("figure.pdf")

sns.violinplot(x="Days", y='Subject', data=data)
plt.savefig("figure2.pdf")

# plot the distribution of Reaction
sns.distplot(data.Reaction)
plt.savefig("figure3.pdf")
plt.show()

# plot the distribution of the days
sns.distplot(data.Days, kde=False)
plt.savefig("figure4.pdf")
plt.show()

sns.lmplot(x = "Days", y = "Reaction", data = data)
plt.savefig("figure5.pdf")

# OLS
modelOLS = smf.ols("Reaction ~ Days", data, groups=data["Subject"])
resultOLS = modelOLS.fit()
print(resultOLS.summary())

# GLM
modelGLM = smf.glm("Reaction ~ Days", data, groups=data["Subject"])
resultGLM = modelGLM.fit()
print(resultGLM.summary())

# LMM
modelLMM = smf.mixedlm("Reaction ~ Days", data, groups=data["Subject"])
resultLMM = modelLMM.fit()
print(resultLMM.summary())

y = data.Reaction
y_predict_LMM = resultLMM.fittedvalues
RMSE_LMM = sqrt(((y-y_predict_LMM)**2).values.mean())
results = pd.DataFrame()
```

```
results["Method"] = ["LMM"]
results["RMSE"] = RMSE_LMM

y_predict_GLM = resultGLM.fittedvalues
RMSE_GLM = sqrt(((y-y_predict_GLM)**2).values.mean())
results.loc[1] = ["GLM",RMSE_GLM]

y_predict_OLS = resultOLS.fittedvalues
RMSE_OLS = sqrt(((y-y_predict_OLS)**2).values.mean())
results.loc[2] = ["OLS",RMSE_OLS]

results

performance = pd.DataFrame()
performance["residuals"] = resultLMM.resid.values
performance["Days"] = data.Days
performance["predicted"] = resultLMM.fittedvalues

sns.lmplot(x = "predicted", y = "residuals", data = performance)

ax = sns.residplot(x = "Days", y = "residuals",
    data = performance, lowess=True)
ax.set(ylabel='Observed - Prediction')
plt.show()
```