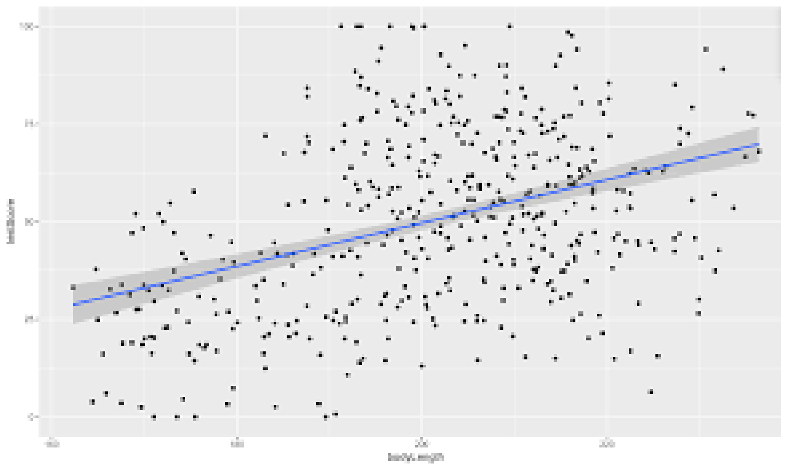
Université Montpellier



Mixed model by REML and Maximum likelihood

DELAGE Cindy
M2 MIND

Teacher : M. SALMON Joseph

8 novembre 2020

Contents

Chapter 1

Passing through the theory

1.1 Mixed Model

1.1.1 General definition

A mixed model is a model that combines two types of effects : fixed effects and random effects. The first ones can be generalized for an entire population, they describe the relation between the covariates and the dependant variable. The second ones are sample-specific.

Model

For linear mixed model :

$$\text{Y = X}\beta \text{ +Zu + } \epsilon$$

$$\begin{pmatrix} u \\ \epsilon \end{pmatrix} \sim (\mathcal{N}(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} G & 0 \\ 0 & R \end{pmatrix}))$$

X and Z are design matrix, β is the fix effect vector and u is the random effect vector.

1.1.2 Linear mixed model by REML and ML

Two types of estimation exist for the mixed models : maximum likelihood (ML) and restricted maximum likelihood (RML). Both use the log likelihood, but ML calculates biased variance components.

Meaning of this article

Using a biological data, we attempted a regression process, using Python, for mixed model, firstly with maximum likelihood estimation and secondly with REML estimation. Every Python code used has been compared to the R version and sometimes the manual estimation.

Chapter 2

Application

The data used for this application comes from [4], it includes three variables (ingot, metal, pres) and 21 individuals. The R code is largely inspired from [2].

2.1 Estimation of the fixed effect vector, β

2.1.1 ML

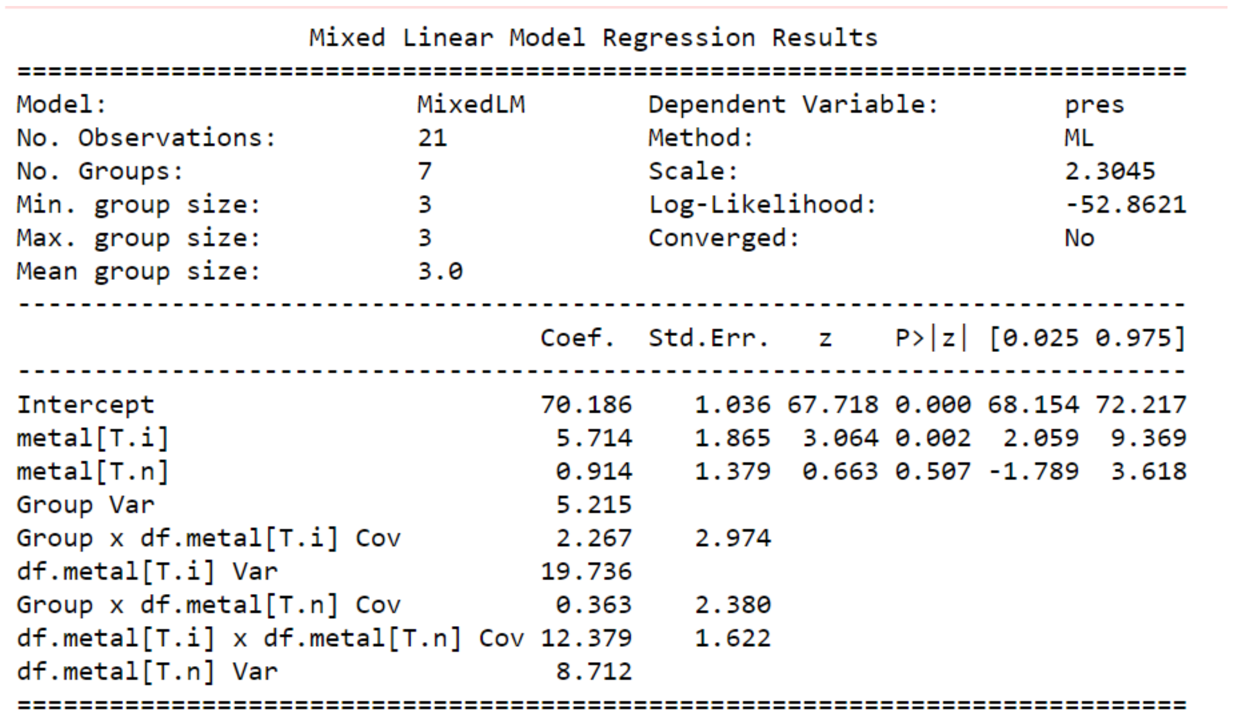
The regression is being made with Y = pres, X = metal and Z = ingot. Our first estimation will be made with the maximum likelihood estimation.

Python

Python uses the function *statsmodels.formula.api.mixedlm* to perform a linear mixed model regression. The code used and the results are presented below.

```
md = smf.mixedlm('pres ~ metal', df, groups=df.ingot, re_formula="~df.metal")
mdf = md.fit(reml=False)
print(mdf.summary())
```

Listing 2.1: ML estimation with Python



```
                    Mixed Linear Model Regression Results
============================================================================
Model:                     MixedLM        Dependent Variable:          pres
No. Observations:          21             Method:                      ML
No. Groups:                7              Scale:                       2.3045
Min. group size:           3              Log-Likelihood:              -52.8621
Max. group size:           3              Converged:                   No
Mean group size:           3.0
----------------------------------------------------------------------------
                                  Coef.  Std.Err.   z    P>|z| [0.025 0.975]
----------------------------------------------------------------------------
Intercept                         70.186    1.036 67.718 0.000 68.154 72.217
metal[T.i]                         5.714    1.865  3.064 0.002  2.059  9.369
metal[T.n]                         0.914    1.379  0.663 0.507 -1.789  3.618
Group Var                          5.215
Group x df.metal[T.i] Cov          2.267    2.974
df.metal[T.i] Var                 19.736
Group x df.metal[T.n] Cov          0.363    2.380
df.metal[T.i] x df.metal[T.n] Cov 12.379    1.622
df.metal[T.n] Var                  8.712
============================================================================
```

The estimated coefficients are listed in the "Coef." column :

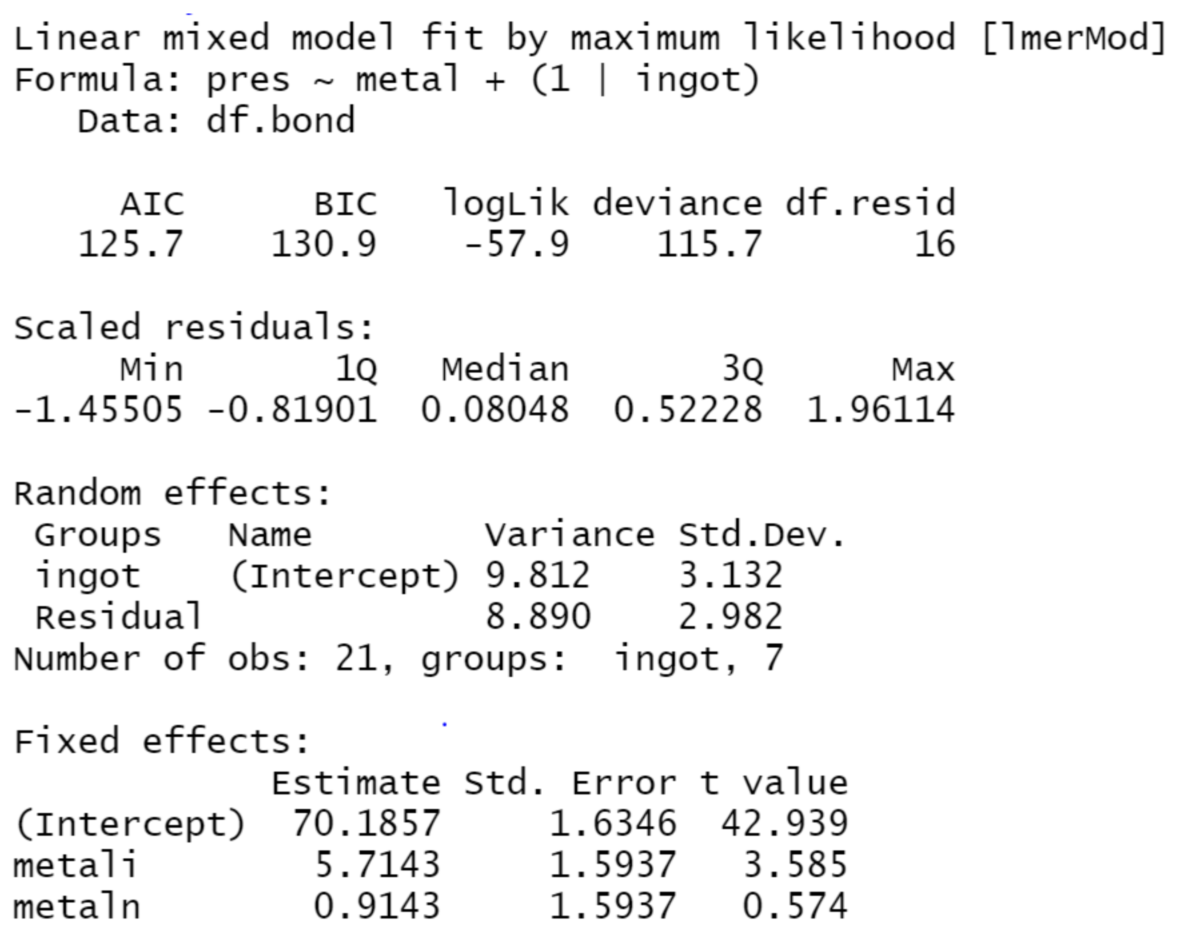
$$\beta = \begin{pmatrix} 70.186 \\ 5.714 \\ 0.914 \end{pmatrix}$$

R version

The function lmer does exactly the same and produces the results below :

```
lmm.bond <- lmer( pres ~ metal + (1|ingot),REML = FALSE,data = df.bond)
summary(lmm.bond)
```

Listing 2.2: ML estimation with R



```
Linear mixed model fit by maximum likelihood [lmerMod]
Formula: pres ~ metal + (1 | ingot)
   Data: df.bond

     AIC      BIC   logLik deviance df.resid
   125.7    130.9    -57.9    115.7       16

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.45505 -0.81901  0.08048  0.52228  1.96114

Random effects:
 Groups   Name        Variance Std.Dev.
 ingot    (Intercept) 9.812    3.132
 Residual             8.890    2.982
Number of obs: 21, groups:  ingot, 7

Fixed effects:
            Estimate Std. Error t value
(Intercept)  70.1857     1.6346  42.939
metali        5.7143     1.5937   3.585
metaln        0.9143     1.5937   0.574
```

Again,

$$\beta = \begin{pmatrix} 70.1857 \\ 5.7143 \\ 0.9143 \end{pmatrix}$$

Python and R give the same coefficients, although R gives a more precise result and has a standard error slightly higher. Let’s finally try a "manual" estimation :

Manual estimation (Python)

For linear mixed model:

$$\hat{\beta} = (X^\top V^{-1} X)^{-1} X^\top V^{-1} Y$$

V is the variance of Y, $V = ZGZ^\top + R$, (R and G have been defined in section 1.1.1).
Using these formulas, we write the Python code :

```
def Beta_estimated(x):
    Y = df.pres
    X = np.array([(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1), (0, 1, 0, 0, 1, 0, 0, 1, 0,

       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0), (1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
    1, 0, 0, 1, 0, 0, 1, 0, 0)])
    Z = np.mat('[1.,1.,1.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.;0.,0.,0.,1.,1.,1.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0.,1.,1.,1.,0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0., 0.; 0., 0., 0., 0., 0., 0., 0., 0., 0.,1.,1.,1.,0., 0.,
   0., 0., 0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0.,1.,1.,1.,0., 0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.,1.,1.,1.,0.,0.,0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0., 0., 0., 0.,1.,1.,1.]')
    G = x[0] * np.identity(7)
    R = x[1] * np.identity(21)
    V = (Z.T@G @ Z)+R
    Beta = np.linalg.inv(X@np.linalg.inv(V)@X.T)@X@np.linalg.inv(V)@Y
    return(Beta)


Beta_estimated(np.array([1, 1]))
```

Listing 2.3: Manual estimation

Running this code gives the following parameters :

$$\beta = \begin{pmatrix} 70.18571429 \\ 5.71428571 \\ 0.91428571 \end{pmatrix}$$

which are the same than the previous ones, with a bigger precision.

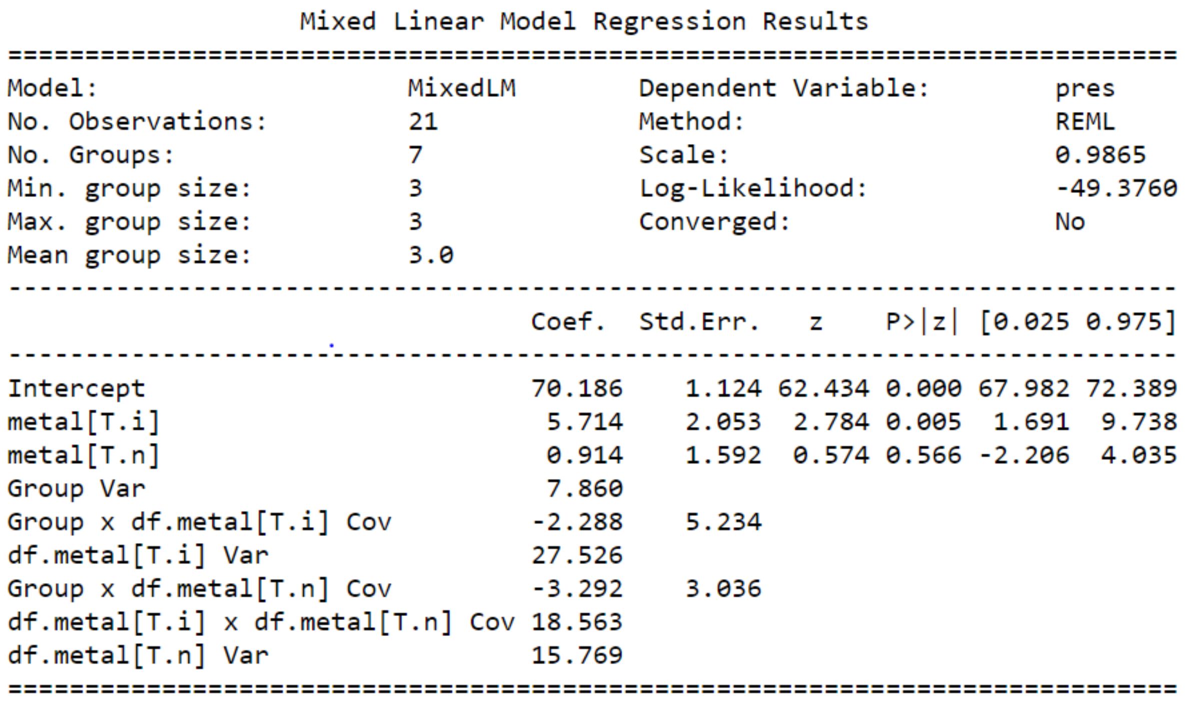
2.1.2 REML

The manual estimation of β does not change with REML estimation, but it's interesting to search for differences in the functions *lmer* and *smf.mixedlm*.

Python

```
md = smf.mixedlm('pres ~ metal', df, groups=df.ingot, re_formula="~df.metal")
mdf = md.fit()
print(mdf.summary())
```

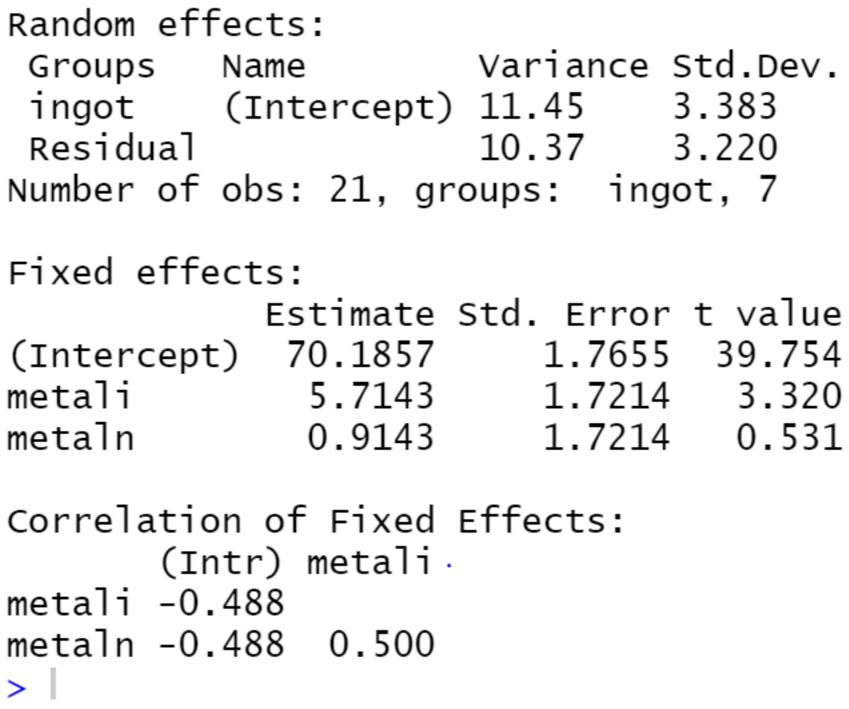
Listing 2.4: REML estimation



```
                      Mixed Linear Model Regression Results
=============================================================================
Model:                      MixedLM         Dependent Variable:        pres
No. Observations:           21              Method:                    REML
No. Groups:                 7               Scale:                     0.9865
Min. group size:            3               Log-Likelihood:            -49.3760
Max. group size:            3               Converged:                 No
Mean group size:            3.0
-----------------------------------------------------------------------------
                                  Coef.  Std.Err.   z    P>|z| [0.025 0.975]
-----------------------------------------------------------------------------
Intercept                         70.186    1.124 62.434 0.000 67.982 72.389
metal[T.i]                         5.714    2.053  2.784 0.005  1.691  9.738
metal[T.n]                         0.914    1.592  0.574 0.566 -2.206  4.035
Group Var                          7.860
Group x df.metal[T.i] Cov         -2.288    5.234
df.metal[T.i] Var                 27.526
Group x df.metal[T.n] Cov         -3.292    3.036
df.metal[T.i] x df.metal[T.n] Cov 18.563
df.metal[T.n] Var                 15.769
=============================================================================
```

```r
lmm.bond <- lmer( pres ~ metal + (1|ingot),REML = TRUE,data = df.bond)
summary(relmm.bond)
```

Listing 2.5: REML estimation



```
Random effects:
 Groups   Name        Variance Std.Dev.
 ingot    (Intercept) 11.45    3.383
 Residual             10.37    3.220
Number of obs: 21, groups:  ingot, 7

Fixed effects:
            Estimate Std. Error t value
(Intercept)  70.1857     1.7655  39.754
metali        5.7143     1.7214   3.320
metaln        0.9143     1.7214   0.531

Correlation of Fixed Effects:
       (Intr) metali
metali -0.488
metaln -0.488  0.500
> 
```

In Python as in R, ML gives smaller standard errors and smaller confidence intervals. We'll try to demonstrate and explain this in the next chapter.

2.2 Estimation of the variance parameters

We will estimate the Y variance and the residuals variance with an optimisation, using the log likelihood function and maximising it. The difference between ML and REML in this case is based on the expression of the likelihood.

2.2.1 ML

Y follows a normal distribution. For this reason, the log likelihood for maximum likelihood estimation is equal to :

$$-2\log(\beta, Y) = log(|V|) + (Y - X\beta)^T V^{-1}(Y - X\beta) + Cste$$

Python

The following code gives the estimation of the variances :

```python
def optim(x):
    Y = df.pres
    X = np.array([(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1), (0, 1, 0, 0, 1, 0, 0, 1, 0,

      0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0), (1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
    1, 0, 0, 1, 0, 0, 1, 0, 0)])
    Z = np.mat('[1.,1.,1.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.;0.,0.,0.,1.,1.,1.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0.,1.,1.,1.,0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0., 0.; 0., 0., 0., 0., 0., 0., 0., 0., 0.,1.,1.,1.,0., 0.,
   0., 0., 0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0.,1.,1.,1.,0., 0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.,1.,1.,1.,0.,0.,0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0., 0., 0., 0.,1.,1.,1.]')
    G = x[0] * np.identity(7)
    R = x[1] * np.identity(21)
    V = (Z.T@G @ Z)+R
    Beta = np.linalg.inv(X@np.linalg.inv(V)@X.T)@X@np.linalg.inv(V)@Y
    y = np.atleast_2d(Y)
    loglike = np.log(np.linalg.det(V)) + (y - (X.T @ Beta)
                                             )@(np.linalg.inv(V)) @ (y-X.T@Beta).T
    return(loglike[0, 0])


minimize(optim, x0=np.array([1, 1]), tol=0.0000000000001)
```

Listing 2.6: RMEL estimation with R

With a tolerance equal to 0.0000000000001 and an initialisation at $x_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, we obtain :

$$\begin{pmatrix} \sigma_Y^2 \\ \sigma_{residuals}^2 \end{pmatrix} = \begin{pmatrix} 9.81236269 \\ 8.88992588 \end{pmatrix}$$

R version

The R code use the same function and gives a result which is pretty close to the one obtained with Python, but this time R has less precision :

```r
loglikef <- function(x) {
  vector.Y <- as.vector(df.bond$pres)
  matrix.G <- x[1] * diag(1,nrow = 7)
  matrix.R <- x[2] * diag(1,nrow = 21)
  matrix.V <- matrix.z %*% matrix.G %*% t(matrix.z) + matrix.R
  vector.Beta <- solve(t(matrix.x) %*% solve(matrix.V) %*% matrix.x) %*% t(
    matrix.x) %*% solve(matrix.V) %*% vector.Y
  loglike <- log(det(matrix.V)) + t(vector.Y - matrix.x %*% vector.Beta) %*%
    solve(matrix.V) %*% (vector.Y - matrix.x %*% vector.Beta)
  return(c(loglike))
}

optim_nm(fun = loglikef, k = 2, start = c(1,1),maximum= FALSE, tol =
    0.0000000000001)$par
```
Listing 2.7: Optimisation

With the same tolerance (equal to 0.0000000000001) and same initialisation at $x_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, we obtain :

$$\begin{pmatrix} \sigma_Y^2 \\ \sigma_{residuals}^2 \end{pmatrix} = \begin{pmatrix} 9.812383 \\ 8.889932 \end{pmatrix}$$

Are these estimations obtained by maximum likelihood close enough to what R guessed with *lmer* ? The summary of the regression with R gives the estimation of the variances for Y and the residuals :

$$\begin{pmatrix} \sigma_Y^2 \\ \sigma_{residuals}^2 \end{pmatrix} = \begin{pmatrix} 9.812 \\ 8.890 \end{pmatrix}$$

Our three estimations are all close to each other.

2.2.2 REML

Python

The only difference in our code between ML and REML will be the log likelihood function : REML adds a term in this function. Using REML, the log likelihood for maximum likelihood becomes :

$$-2\log(\beta, Y) = log(|V|) + \log(\left|X^T V{-}1X\right|) + (Y - X\beta)^T V^{-1}(Y - X\beta) + Cste$$

```
def optim_REML(x):
    Y = df.pres
    X = np.array([(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1), (0, 1, 0, 0, 1, 0, 0, 1, 0,

       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0), (1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
    1, 0, 0, 1, 0, 0, 1, 0, 0)])
    Z = np.mat('[1.,1.,1.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.;0.,0.,0.,1.,1.,1.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0.,1.,1.,1.,0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0., 0.; 0., 0., 0., 0., 0., 0., 0., 0., 0.,1.,1.,1.,0., 0.,
   0., 0., 0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0.,1.,1.,1.,0., 0., 0., 0., 0., 0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0.,1.,1.,1.,0.,0.,0.;0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
   0., 0., 0., 0., 0., 0., 0., 0.,1.,1.,1.]')
    G = x[0] * np.identity(7)
    R = x[1] * np.identity(21)
    V = (Z.T@G @ Z)+R
    Beta = np.linalg.inv(X@np.linalg.inv(V)@X.T)@X@np.linalg.inv(V)@Y
    y = np.atleast_2d(Y)
    loglike = np.log(np.linalg.det(V)) + np.log(np.linalg.det(X@np.linalg.inv(V)
                                                             @ X.T)) + (y - (X.
   T @ Beta))@(np.linalg.inv(V)) @ (y-X.T@Beta).T
    return loglike[0, 0]


minimize(optim_REML, x0=np.array([5, 6]), tol=0.0000000000001)
```
Listing 2.8: REML optimisation

We obtain :

$$\begin{pmatrix} \sigma_Y^2 \\ \sigma_{residuals}^2 \end{pmatrix} = \begin{pmatrix} 11.44780323 \\ 10.3715852 \end{pmatrix}$$

Again, R, whether we use the manual estimation or *lmer*, gives the same estimations but with less precision. The estimators are bigger with REML than with ML, we'll explain this mathematically in the following chapter.

Chapter 3

Back to theory : Differences between REML and ML

The application of the linear mixed model regression highlighted two major differences between REML and ML estimations :

- First, the confidence intervals with ML are smaller.
- Secondly, the estimated variances are higher with REML. All of these differences are explained by one property of the ML estimator : it has a bias.

Let's demonstrate it with σ_Y^2, that we will name σ^2 for a question of legibility. The following proof is inspired from [3].

Reminder : $Y \sim \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(X\beta, ZGZ^T + R)$

$$\begin{aligned}
\hat{\sigma^2} &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}((y_i - \mu) - (\hat{y} - \mu))^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 - \frac{2}{N}\sum_{i=1}^{N}(y_i - \mu)(\hat{y} - \mu) + \frac{1}{N}\sum_{i=1}^{N}(\hat{y} - \mu)^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 - \frac{2(\hat{y} - \mu)}{N}\sum_{i=1}^{N}(y_i - \mu) + (\hat{y} - \mu)^2
\end{aligned}$$

Besides,

$\hat{y} - \mu = \frac{1}{N}\sum_{i=1}^{N} y_i - \frac{1}{N}\sum_{i=1}^{N}\mu = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)$

$\implies \sum_{i=1}^{N}(y_i - \mu) = N(\hat{y} - \mu)$

Thus,

$\hat{\sigma^2} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 - \frac{2(\hat{y}-\mu)}{N}\sum_{i=1}^{N}(y_i - \mu) + (\hat{y} - \mu)^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 - (\hat{y} - \mu)^2$

Finally, let's calculate the estimated mean :

$E(\hat{\sigma^2}) = E[\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2] - E[(\hat{y} - \mu)^2] = \sigma^2 - E[(\hat{y} - \mu)^2]$

$E[(\hat{y} - \mu)^2] = V[\hat{y}] = \frac{1}{N^2}\sum_{i=1}^{N} V[y_i] = \frac{\sigma^2}{N}$
We come to the conclusion :

$E[\hat{\sigma}^2] = \sigma^2 - \frac{\sigma^2}{N}$

We can see that the estimated $\hat{\sigma}^2$ has a bias. More precisely, it's a downward bias, which means that the variance obtained is, in terms of mean, smaller than the real variance. REML corrects this bias by adding a term to the log likelihood, which is why the REML estimation of the variance is closer to the real variance and bigger than the ML one, which underestimates the reality. In fact, for each variance estimated by ML (not only for Y but for the residuals and for β), the result will be underestimated, it can be proved the same way as for ${\sigma_Y}^2$.
Besides, the ML confidence interval will be smaller because the variance is used to calculate the CI :

CI = [$\hat{\beta}$ -/+ $c * {\sigma_{\hat{beta}}}^2$], where c is a constant. The smaller the variance is, the smaller the confidence interval.

Chapter 4

Conclusion

With Python, we reproduced a R code allowing us to perform a linear mixed model regression, firstly with a maximum likelihood regression and secondly with a restrained maximum likelihood regression. We studied the main differences between the RML and ML estimations for linear mixed model. The confidence intervals for maximum likelihood were smaller than for restrained maximum likelihood, so was the standard deviation. All of these differences are due to the same property of the ML estimator : it is a biased estimator.

Bibliography

[1] Nikolay Oskolkov. *Maximum Likelihood (ML) vs. REML.* towardsdatascience. 2020. `https://towardsdatascience.com/maximum-likelihood-ml-vs-reml-78cf79bef2cf`

[2] Kevin Liu. *Example for Mixed Model Equations and REML.* (Github) 2018. `https://rh8liuqy.github.io/Example`$_{L}inear_{m}ixed_{m}odel.html$

[3] Kevin Liu. *Some important proof of Mixed Model Equations and REML.* (Github) 2018. `https://rh8liuqy.github.io/Linear`$_{m}ixed_{m}odel_{e}quations.html$

[4] Littell, Ramon C., George A. Milliken, Walter W. Stroup, Russell D. Wolfinger, and Schabenberber Oliver. *SAS for Mixed Models, Second Edition.*