Chapter 1

A Simplified LME Framework for Method Comparison

1.1 Systolic Blood Pressure Data Set

? provides three case studies, using data sets well known in method comparison studies, to demonstrate how the methodology should be used. The first two examples used are from the 'blood pressure' data set introduced by **?**. The data set is a tabulation of simultaneous measurements of systolic blood pressure were made by each of two experienced observers (denoted 'J' and 'R') using a sphygmomanometer and by a semi-automatic blood pressure monitor (denoted 'S'). Three sets of readings were made in quick succession. Roy compares the 'J' and 'S' methods in the first of her examples.

1.2 Implementation in `R`

To implement an LME model in `R`, the `nlme` package is used. This package is loaded into the `R` environment using the library command, (i.e. `library(nlme)`). The `lme` command is used to fit LME models. The first two arguments to the `lme` function specify the fixed effect component of the model, and the data set to which the model is to be fitted. The first candidate model ('`ref.nlme`') fits an LME model on the data set 'Blood'. The variable '`meth`' is assigned as the fixed effect, with the response variable '`y`' (i.e. blood pressure).

The third argument contain the random effects component of the formulation, describing the random effects, and their grouping structure. The `nlme` package provides a set of positive-definite matrices , the `pdMat` class, that can be used to specify a structure for the between-subject variance-covariance matrix for the random effects. For Roy's models, we will use the `pdSymm` and `pdCompSymm` to specify a symmetric structure and a compound symmetry structure respectively. A full discussion of these structures can be found in **?**, pg. 158.

Similarly a variety of structures for the with-subject variance-covariance matrix can be implemented using `nlme`. To implement a particular matrix structure, one must specify both a variance function and correlation structure accordingly. Variance functions are used to model the variance structure of the within-subject errors. `varIdent` is a variance function object used to allow different variances according to the levels of a classification factor in the data. A compound symmetry structure is implemented using the `corCompSymm` class, while the symmetric form is specified by `corSymm` class. Finally, the estimation methods is specified as "ML" or "REML".

`y` : Response variable

`meth` : Method of Measurement

`item` : Subject

`repl` : Replicate index

Using this `R` implementation for other data sets requires that the data set is structured appropriately (i.e. each case of observation records the index, response, method and replicate). Once formatted properly, implementation is simply a case of re-writing the first line of code, and computing the four candidate models accordingly.

1.3 Implementation in R

To implement an LME model in R, the nlme package is used. This package is loaded into the R environment using the library command, (i.e. library(nlme)). The lme command is used to fit LME models. The first two arguments to the lme function specify the fixed effect component of the model, and the data set to which the model is to be fitted. The first candidate model ('ref.nlme') fits an LME model on the data set 'Blood'. The variable 'meth' is assigned as the fixed effect, with the response variable 'y' (i.e. blood pressure).

The third argument contain the random effects component of the formulation, describing the random effects, and their grouping structure. The nlme package provides a set of positive-definite matrices , the pdMat class, that can be used to specify a structure for the between-subject variance-covariance matrix for the random effects. For Roy's models, we will use the pdSymm and pdCompSymm to specify a symmetric structure and a compound symmetry structure respectively. A full discussion of these structures can be found in **?**, pg. 158.

Similarly a variety of structures for the with-subject variance-covariance matrix can be implemented using nlme. To implement a particular matrix structure, one must specify both a variance function and correlation structure accordingly. Variance functions are used to model the variance structure of the within-subject errors. varIdent is a variance function object used to allow different variances according to the levels of a classification factor in the data. A compound symmetry structure is implemented using the corCompSymm class, while the symmetric form is specified by corSymm class. Finally, the estimation methods is specified as "ML" or "REML".

y : Response variable

meth : Method of Measurement

item : Subject

repl

Using this R implementation for other data sets requires that the data set is structured appropriately (i.e. each case of observation records the index, response, method and replicate). Once formatted

properly, implementation is simply a case of re-writing the first line of code, and computing the four candidate models accordingly.

1.4 Computation of LMEs using R

? advises how to implement LME models in statistical software (ostensibly for S and S PLUS, but `R` is very similar). When tackling linear mixed effects models using the R language, a statistician can call upon the `lme` command found in the `nlme` package. This command fits a LME model to the data set using either Maximum Likelihood (ML) or Restricted Maximum Likelihood (REML).

The first two arguments for `lme` are `fixed` and `data`, which give the model for the expected responses (i.e. the fixed part of the model), and the data that the model should be fitted from. The next argument is *random*, a one-sided formula which describes the random effects, and the grouping structure for the model. The *method* argument can specify whether to use 'REML', the default setting, or 'ML'.

1.5 Important Consideration for MCS

The key issue is that `nlme` allows for the particular specification of Roy's Model, specifically direct specification of the VC matrices for within subject and between subject residuals.

The `lme4` package does not allow for Roy's Model, for reasons that will identified shortly.To advance the ideas that eminate from Roy's paper, one is required to use the `nlme` context. However, to take advantage of the infrastructure already provided for `lme4` models, one may change the research question away from that of Roy's paper. To this end, an exploration of what ***influence.ME*** can accomplished is merited. The first of Roy's candidate model can be implemented using the following code;

```
ref.nlme = lme(BP ~ method-1, data = dat,
random =  list(subject=pdSymm(~ method-1)),
weights=varIdent(form=~1|method),
```

```
correlation = corSymm(form=~1 | subject/obs), method="ML")
```

For the blood pressure data used in **?**, all four candidate models are implemented by slight variations of this piece of code, specifying either `pdSymm` or `pdCompSymm` in the second line, and either `corSymm` or `corCompSymm` in the fourth line. For example, the second candidate model 'MCS2' is implemented with the same code as MCS1, except for the term `pdCompSymm` in the second line, rather than `pdSymm`.

```
    test1.nlme = lme(BP ~ method-1, data = dat,
random = list(subject=pdCompSymm(~ method-1)),
weights = varIdent(form=~1|method),
correlation = corSymm(form=~1 | subject/obs), method="ML")
```

The fixed effects estimates are the same for all four candidate models. The inter-method bias can be easily determined by inspecting a summary of any model. The summary presents estimates for all of the important parameters, but not the complete variance-covariance matrices (although some simple `R` functions can be written to overcome this). The variance estimates for the random effects for `test1.nlme` is presented below.

```
Random effects:
Formula: ~method - 1 | subject
Structure: Compound Symmetry
StdDev Corr
methodJ  30.765
methodS  30.765 0.829
Residual  6.115
```

Similarly, for computing the limits of agreement the standard deviation of the differences is not explicitly given. Again, a simple R function can be written to calculate the limits of agreement directly.

1.6 Computation of LMEs using R

? advises how to implement LME models in statistical software (ostensibly for S and S PLUS, but R is very similar). When tackling linear mixed effects models using the R language, a statistician can use the lme() function, from the nlme package.

The first two arguments for lme are fixed and data, which give the model for the expected responses (i.e. the fixed part of the model), and the data that the model should be fitted from. The next argument is *random*, a one-sided formula which describes the random effects, and the grouping structure for the model.

The lme() command fits a LME model to the data set using either Maximum Likelihood (ML) or Restricted Maximum Likelihood (REML). The *method* argument can specify whether to use 'REML', the default setting, or 'ML'.

1.7 Important Consideration for Method Comparison

The key issue is that nlme allows for the particular specification of Roy's Model, specifically direct specification of the VC matrices for within subject and between subject residuals.

The lme4 package does not allow for Roy's Model, for reasons that will identified shortly.To advance the ideas that eminate from Roy's paper, one is required to use the nlme context. However, to take advantage of the infrastructure already provided for lme4 models, one may change the research question away from that of Roy's paper. To this end, an exploration of what ***influence.ME*** can accomplished is merited. The first of Roy's candidate model can be implemented using the following code;

```
ref.nlme <- lme(BP ~ method-1, data = dat,
random =  list(subject=pdSymm(~ method-1)),
weights=varIdent(form=~1|method),
correlation = corSymm(form=~1 | subject/obs),
method="ML")
```

For the blood pressure data used in **?**, all four candidate models are implemented by slight variations of this piece of code, specifying either `pdSymm` or `pdCompSymm` in the second line, and either `corSymm` or `corCompSymm` in the fourth line. For example, the second candidate model 'MCS2' is implemented with the same code as MCS1, except for the term `pdCompSymm` in the second line, rather than `pdSymm`.

```
test1.nlme <- lme(BP ~ method-1, data = dat,
random = list(subject=pdCompSymm(~ method-1)),
weights = varIdent(form=~1|method),
correlation = corSymm(form=~1 | subject/obs),
method="ML")
```

The fixed effects estimates are the same for all four candidate models. The inter-method bias can be easily determined by inspecting a summary of any model. The summary presents estimates for all of the important parameters, but not the complete variance-covariance matrices (although some simple `R` functions can be written to overcome this). The variance estimates for the random effects for `test1.nlme` is presented below.

```
Random effects:
Formula: ~method - 1 | subject
Structure: Compound Symmetry
StdDev Corr
methodJ  30.765
methodS  30.765 0.829
Residual  6.115
```

Similarly, for computing the limits of agreement the standard deviation of the differences is not explicitly given. Again, a simple `R` function can be written to calculate the limits of agreement directly.

1.8 Fitting Models with the LME4 R package

Two LME models are fitted to the data, one using the nlme package, one with the lme4 package. These models shall be called "`ref.nlme`" and "`ref.lme4`" respectively.

In both cases the method is characterized by a fixed effect, while there is a random effect for each subject. This random effect accounts for the replicate measurements associated with each subject. The differences between the estimate provided by the respective models are negligible, due to the simplicity of the model specification.

Maximum likelihood or restricted maximum likelihood (REML) estimates of the parameters in linear mixed-effects models can be determined using the `lmer` function in the ***lme4*** package for `R`. As for most model-fitting functions in `R`, the model is described in an `lmer` call by a formula, in this case including both fixed- and random-effects terms.

The formula and data together determine a numerical representation of the model from which the profiled deviance or the profiled REML criterion can be evaluated as a function of some of the model parameters. The appropriate criterion is optimized, using one of the constrained optimization functions in `R`, to provide the parameter estimates. We describe the structure of the model, the steps in evaluating the profiled deviance or REML criterion, and the structure of classes or types that represents such a model.

Sufficient detail is included to allow specialization of these structures by users who wish to write functions to fit specialized linear mixed models, such as models incorporating pedigrees or smoothing splines, that are not easily expressible in the formula language used by `lmer()`.

1.9 Fitting Models with the LME4 R package

Two LME models are fitted to the data, one using the nlme package, one with the lme4 package. These models shall be called "`ref.nlme`" and "`ref.lme4`" respectively.

In both cases the method is characterized by a fixed effect, while there is a random effect for each subject. This random effect accounts for the replicate measurements associated with each subject. The differences between the estimate provided by the respective models are negligible, due to the simplicity of the model specification.

Maximum likelihood or restricted maximum likelihood (REML) estimates of the parameters in linear mixed-effects models can be determined using the `lmer` function in the ***lme4*** package for `R`. As for most model-fitting functions in `R`, the model is described in an `lmer` call by a formula, in this case including both fixed- and random-effects terms.

The formula and data together determine a numerical representation of the model from which the profiled deviance or the profiled REML criterion can be evaluated as a function of some of the model parameters. The appropriate criterion is optimized, using one of the constrained optimization functions in `R`, to provide the parameter estimates. We describe the structure of the model, the steps in evaluating the profiled deviance or REML criterion, and the structure of classes or types that represents such a model.

Sufficient detail is included to allow specialization of these structures by users who wish to write functions to fit specialized linear mixed models, such as models incorporating pedigrees or smoothing splines, that are not easily expressible in the formula language used by `lmer()`.