Manual for influence.ME: Tools for detecting influential data in mixed models Version 0.7

Rense Nieuwenhuis[1][2], Ben Pelzer[3], Manfred te Grotenhuis[4]

July 9, 2009

[1]Corresponding author: contact@rensenieuwenhuis.nl
[2]University of Twente, Enschede
[3]Radboud University, Nijmegen.
[4]Radboud University, Nijmegen.

Contents

Chapter 1

Introduction

Applying mixed effects regression models tends to become common practice in the field of social sciences. In the social sciences, these models often are refered to as multilevel models, and entail observations (eg. respondents) nested within higher-level units or groups (eg. countries, states, or schools). However, diagnostic tools to evaluate these models lag behind. For instance, there is no general applicable tool to check whether all units (or cases) roughly have the same influence on the regression parameters. It is however commonly accepted that tests for influential cases should be performed, especially when estimates are based on a relatively small number of cases (Van der Meer et al., 2009).

Testing for (overly) influential cases within mixed effects models is especially important in social science applications, for two reasons. First, models in the social sciences are frequently based on large numbers of individuals while the number of higher level units often is relatively small. For instance, it was found that no country-comparative study exceeds data nested within 54 countries (Van der Meer et al., 2009). Secondly, often the higher level units are remarkably similar, for instance in the case of neighboring countries. Therefore, not just single higher level units should be evaluated, but multiple as well. Influence.ME is a package for R that provides two innovations for evaluating influential cases: it extends existing procedures to mixed effects models[1], and it allows to not only search for single influential cases, but for combinations of cases that as a combination exert too much influence, as well.

The basic rationale behind measuring influential cases is that when iteratively single units are omitted from the data, models based on these data should not produce substantially different estimates. To standardize the assessment of how influential a single observation is, several measures of influence are common practice. First, DFBETAS is a standardized measure of the absolute difference between the estimate with a particular case included and the estimate without that particular case (Belsley et al., 1980). Second, Cook's distance provides an overall measurement of the change in all parameter estimates, or a selection

[1] influence.ME works in conjunction with mixed effects models estimated with the `lme4` packages (Bates et al., 2008)

thereof (Cook, 1977). To apply the same logic to mixed effects models one has to measure the influence of a particular higher level unit on the estimates of a higher level predictor. This means that the mixed effect model has to be adjusted to neutralize the unit's influence on that estimate, while at the same time allowing the unit's lower-level cases to help estimate the effects of the lower-level predictors in the model. This procedure is based on a modification of the intercept and the addition of a dummy variable for the cases that might be influential (Langford and Lewis, 1998).

1.1 Preliminaries to this manual

This manual is aimed at those who have basic knowledge of using R, including on how to estimate mixed effects regression models using the `lme4` package (Bates et al., 2008). The use of the `influence.ME` package to detect influential data in mixed effects models will be exemplified in three chapters, using analyses on data described in the section below. The first introduction to influence.ME, in the next chapter, will provide a cursory overview of how the analysis using `influence.ME` takes place. Most important aspects of functionality will be shown, but not all details are discussed. More detail is provided, combined with a description of the internal workings of the package, in the third chapter of this manual. This chapter also gives some information on installing extension packages in R and on getting help for those unfamiliar with R. Finally, in the fourth, chapter, very detailed information on all functions is given. This last chapter is a reprint of the information found in the help-pages of the `influence.ME` package within R.

1.1.1 The school23 Example

The school23 data contains information on students' performance on a math test, as well as several explanatory variables. The explanatory variables are both at the level of the student, as on the level of the school. Student's class and school are equivalent in this data, in the sense that only one class per school in available. These data are a subset of the NELS-88 data (National Education Longitudinal Study of 1988) (Kreft and De Leeuw, 1998). Both a selected number of variables and a selected number of observations are provided within the `influence.ME` package. These data are used in the examples given in Kreft & De Leeuw. The examples and the data are publicly available from the internet: http://www.ats.ucla.edu/stat/examples/imm/. Data are reproduced with kind permission of Ita Kreft and Jan de Leeuw.

Users can examine the school23 data.frame in R using the following commands, when influence.ME is installed:

```
library(influence.ME)
data(school23, package="influence.ME")
str(school23)
```

Based on the syntax above, the package `influence.ME` is loaded, as well as two other packages that are required for `influence.ME` to work properly: `lme4` Bates et al. (2008) and `lattice` Sarkar (2008). The `str()` command returns the structure of the data.frame `school23`.

Chapter 2

A Basic Run-Through of influence.ME

This chapter will provide a cursory introduction to the `influence.ME` package, aimed at users already familiar with estimating mixed models in R. Using the `school23` dataset, a mixed model will be estimated, which subsequently shall be evaluated. The most important functions in the `influence.ME` package will be discussed and demonstrated, but not all. Next, in the third chapter, the internal workings of the functions are described. Detailed information on each specific function will be provided in the fourth and final chapter of this manual, which contains a reprint of most of the information also available in the help pages of the package. These pages therefore can also be accessed from within R.

2.1 Data and Model

The `school23` data contains information on a math test performance of 519 students, who are nested within 23 schools. For this example, we will be interested in the relationship between class structure (in this data measured at the school level) and students' performance on a math test. The research question is: To what extend does the classroom structure determine the students' math test outcomes?

Initially, we will estimate the effect of class `structure` on the result of the `math` performance test, without any further covariates. We do take into account the nesting structure of the data, however, and allow the intercept to be random over schools. This model is estimated using the following syntax, and is assigned to an object we call '`model`'.[1]

[1]Note, that since the model is estimated using the lmer function from the lme4 package, normally this package should be called for using `library(lme4)`, before such an analysis can be performed. It is, however, assumed here that the examination of the `school23` data in the previous chapter already has taken place. When the `influence.ME` package is activated using

```
model <- lmer(math ~  structure + (1 | school.ID), data=school23)
summary(model)
```

The call for a summary of the model results in the output shown below. In this summary, the original model formula is shown, as well as the data on which this model was estimated. Both random and fixed effects are summarized. The amount of intercept variance associated with the nesting structure of students within schools is considerably large (23.8 compared with 81.2 + 23.8 = 104 in total). The effect of interest is that of the structure variable, which is -2.343 and statistically insignificant by most reasonable standards (t=-1.609).

```
Linear mixed model fit by REML
Formula: math ~ structure + (1 | school.ID)
   Data: school23
  AIC  BIC logLik deviance REMLdev
 3802 3819  -1897     3798    3794
Random effects:
 Groups    Name        Variance Std.Dev.
 school.ID (Intercept) 23.884   4.8871
 Residual              81.270   9.0150
Number of obs: 519, groups: school.ID, 23
Fixed effects:
            Estimate Std. Error t value
(Intercept)   60.002      5.853  10.252
structure     -2.343      1.456  -1.609
Correlation of Fixed Effects:
          (Intr)
structure -0.982
```

2.2 Visual Examination

Since the analysis in the previous section has been based on only 23 schools, it is, of course, possible that observations on single schools have overly influenced these findings. Before using the tools provided in the `influence.ME` package to evaluate this, a visual examination of the relationship between class structure and math test performance, aggregated to the school level, will be persued. This visual examination is based solely on tools provided in the base R installation.

```
agg.struct <- tapply(school23$structure, school23$school.ID, mean)
agg.math <- tapply(school23$math, school23$school.ID, mean)
```

Using the `tapply()` function, the mean score on the `structure` variable is calculated for each school, and the resulting vector is assigned to an object

`library(influence.ME)`, the `lme4` package is automatically activated as well.

named `agg.struct`. The same is done for the `math` variable, which is attibuted to the `agg.math` object.

```
par(mfrow=c(1,2))
plot(agg.struct, agg.math,
    xlab="Level of Class Structure",
    ylab="Average Math Test Score",
    main="Math score by Class Structure, by school")
plot(agg.struct, agg.math, type="n",
    xlab="Level of Class Structure",
    ylab="Average Math Test Score",
    main="Math score by Class Structure, by school")
text(agg.struct,agg.math,labels=names(agg.math), cex=.7)
par(mfrow=c(1,1))
```

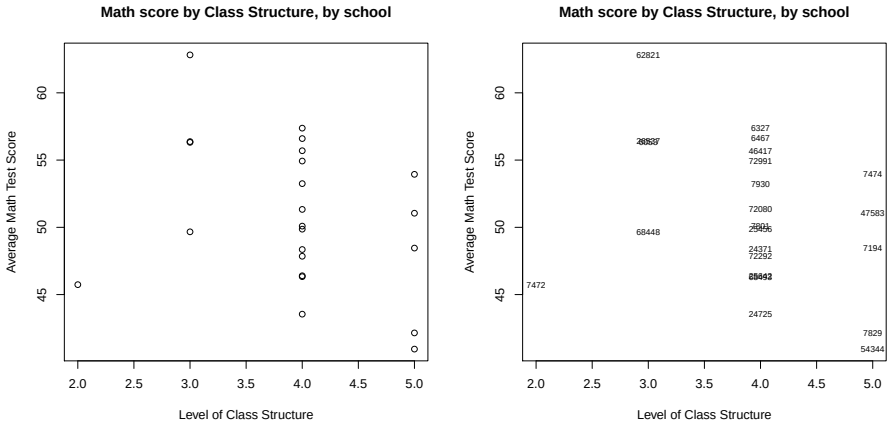
In the syntax above, the `agg.struct` and `agg.math` objects are used to create two graphs, shown in Figure 2.1. The command `par(mfrow=c(1,2))` is used to assure that two graphs are placed together in the same plot window (at the end of the syntax this is re-set to 1). The first `plot()` command just plots the aggregated math test score against the aggregated class structure. Using `xlab=`, `ylab=`, and `main=`, labels are set. This results in the bivariate plot shown in Figure 2.1 (left panel). In this plot, it clear immediately, that the single school represented in the lower-left corner of the graph seems to be an outlier, and may very well have enough leverage to draw a regression line towards itself. It remains unclear from this graph, however, which school this is.

Hence, a second graph is made, plotting the names of the `agg.math` vector instead of circles as the plotting character. Because the `agg.math` vector has been produced using the `tapply()` function, the names of the `agg.math` vector correspond to the labels indicating the schools in the `school.ID` variable. This second graph is created in two steps: first, the plotting region is set up using syntax highly similar to that used for the original graph, but now an additional option is given: `type=''n''`. This way, the plotting region is set up, but no actual points are plotted. Instead, using the `text()` function, the names of the `agg.math` vector are plotted on the appropriate coordinates, resulting in the right panel of Figure 2.1. Based on this Figure, it can be expected that the combination of the low average math test results of students in schools 7472, and the low level of class structure in that school, may have overly influenced the analysis shown in section 2.1.

2.3 Three Steps in influence.ME Procedure

To formally test to what extend observations nested within a single grouping factor have overly influenced a mixed effects model, several standardized measures of influence are available. As was indicated in the introduction, the basic

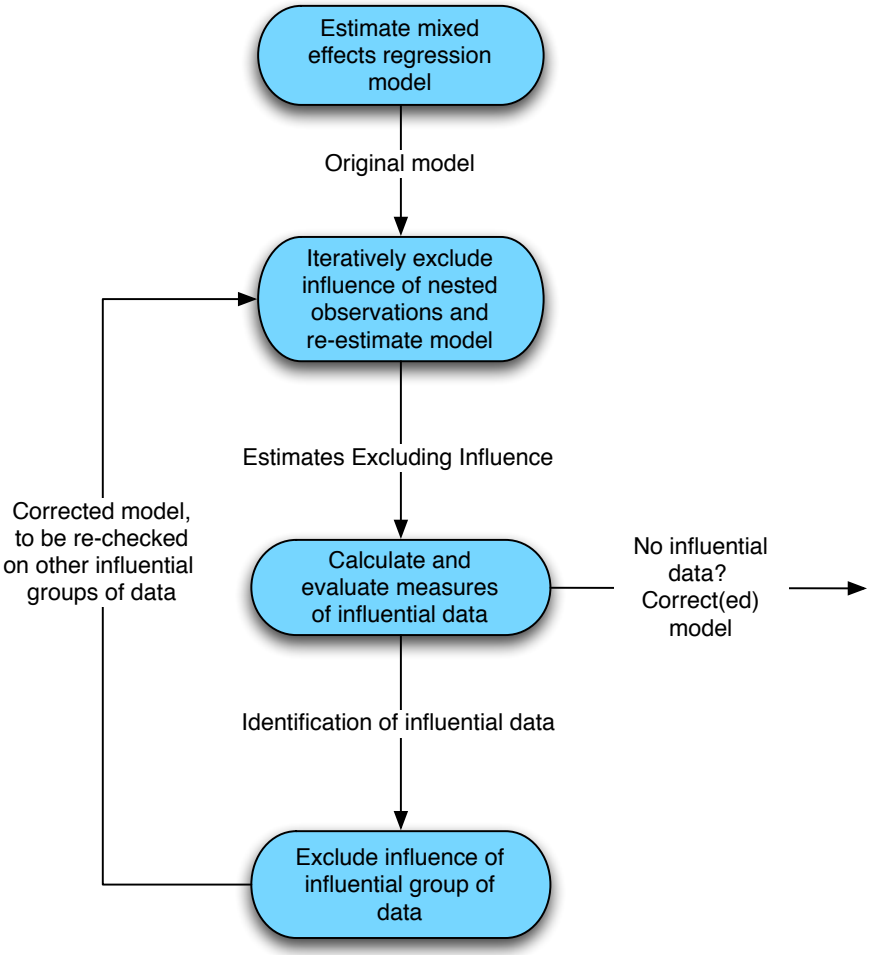
Figure 2.1: Bivariate influence plots



rationale behind measuring influential cases is that when iteratively single units are omitted from the data, models based on these data should not produce substantially different estimates. To standardize the assessment of how influential a single observation is, several measures of influence are common practice. First, DFBETAS is a standardized measure of the absolute difference between the estimate with a particular case included and the estimate without that particular case (Belsley et al., 1980). Second, Cook's distance provides an overall measurement of the change in all parameter estimates, or a selection thereof (Cook, 1977; Snijders and Bosker, 1999; Snijders and Berkhof, 2008). To apply the same logic to mixed effects models one has to measure the influence of a particular higher level unit on the estimates of a higher level predictor. This means that the mixed effect model has to be adjusted to neutralize the unit's influence on that estimate, while at the same time allowing the unit's lower-level cases to help estimate the effects of the lower-level predictors in the model.

This rationale is reflected and executed in the `influence.ME` package. After a mixed effects model has been estimated using the `lme4` package, a typical analysis using the `influence.ME` package entails three steps, which can be repeated as often as necessary. The first step is to iteratively exclude the influence that observations nested within single grouping factors have on the outcomes of the analysis, and to re-estimate the mixed model without that influence. The new model parameters are then used to compute measures of influence, such as Cook's distance and DFBETAS. If the evaluation of these measures leads to the conclusion that indeed overly influential groups of observations are present, their influence needs to be excluded from the analyses. Of course, after the exclusion of the influence of such a group of observations, other groups of observations may still be too influential, and thus these three steps can be repeated untill the user is confident about the stability of the model. These three steps are schematized in Figure 2.2.

Figure 2.2: Three steps in Influence.ME procedure



2.3.1 Iteratively Re-estimate Model

Building upon the example model estimated in section 2.1, the first step in the procedure of the `influence.ME` package is to iteratively exclude the influence of the observations nested within each school separately. This is done using the `estex()` function. The name estex refers to the ESTimates that are returned while EXcluding the influence of each of the grouping levels separately. Thus, in the case of the math test example, in which students are nested in 23 schools, the estex procedure re-estimates the original model 23 times, excluding the influence of a higher level unit (ie school). The function returns the relevant estimates of these 23 re-estimations, which in Figure 2.2 is referred to with 'altered estimates'.

The `estex()` function requires the specification of two parameters: a mixed effects model is to be specified, and the grouping factor of which the influence of the nested observations are to be evaluated.[2] In the syntax example below, the original object '`model`' is specified, and '`school.ID`' is the relevant grouping factor. `school.ID` is the name of the variable used to indicate the grouping factor when the original model was specified. The `estex()` function works perfectly when more than a single grouping is present in the model, but only one grouping factor can be addressed at once.

In the example below, the estimates excluding the influence of the respective grouping levels, as returned by the `estex()` function, are assigned to an object, which in this case is called `este.model` (the name of this object, however, is to be chosen arbitrarily by the user).

```
estex.model <- estex(model, "school.ID")
```

Note that in the case of complex mixed models (i.e. models with large numbers of observations, complex nesting structures, and/or many nesting groups) the execution of `estex()` may consume considerable amounts of time. The examples offered by the `school23` data, should offer no such problems, however.

2.3.2 Calculate Measures of Influence

The object `estex.model` containing the altered estimates can be used to calculate several measures of influential data. To determine the Cook's distance, the `ME.cook()` function is to be used. In its most basic specification, the ME.cook() function only requires an object to which the altered estimates as returned by the estex() function were assigned:

```
ME.cook(estex.model)
```

This basic specification returns a matrix with the rows representing the groups in which the observations are nested, and the single column represents the associated value of Cook's distance. Clearly, these can also be assigned to an object

[2] More options are available in this, and other, functions. Refer to the third chapter of this manual for a more detailed specification of these options.

for later modification. The output below shows the result of the syntax above, representing the Cook's distance associated with each school in the `school23` data.

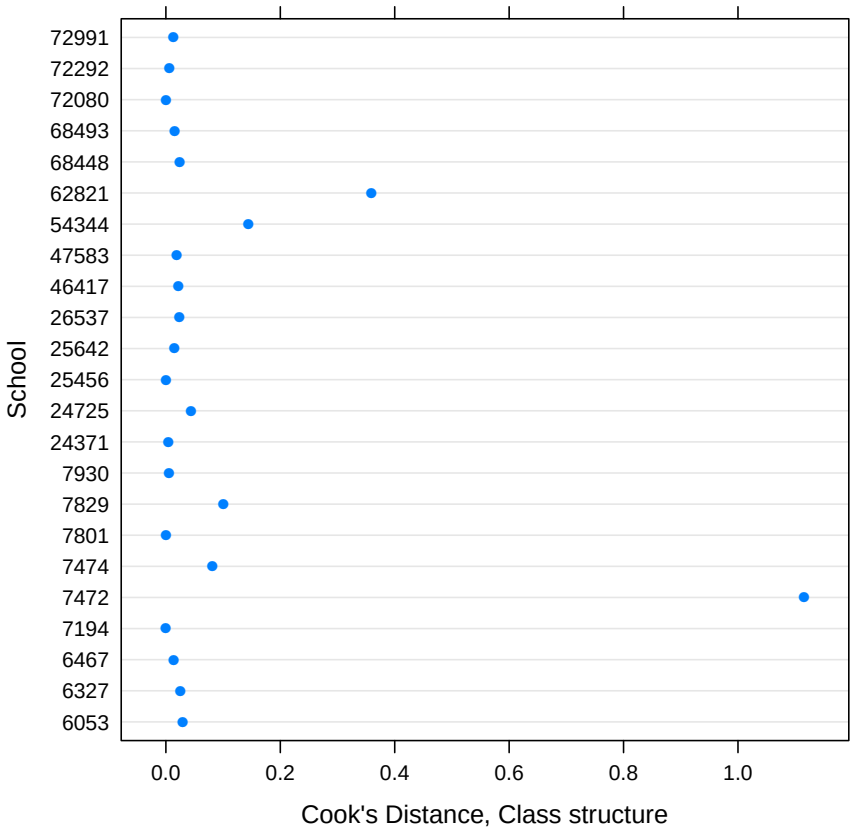
```
              [,1]
6053  2.927552e-02
6327  2.557810e-02
6467  1.402948e-02
7194  3.443392e-05
7472  1.115626e+00
7474  8.142758e-02
7801  3.007558e-04
7829  1.005329e-01
7930  5.525680e-03
24371 4.334659e-03
24725 4.387907e-02
25456 5.644399e-04
25642 1.470130e-02
26537 2.369898e-02
46417 2.204840e-02
47583 1.891108e-02
54344 1.445087e-01
62821 3.593314e-01
68448 2.427028e-02
68493 1.538479e-02
72080 3.471805e-04
72292 6.387956e-03
72991 1.316049e-02
```

Based on the output shown above, the Cook's distance of school number 7472 is the largest. This corresponds very well to what was concluded based on Figure 2.1. For those who prefer to evaluate the Cook's distance based on a visual representation, the `ME.cook()` function can also plot its output.[3] To do so, an additional parameter is required, stating `plot=TRUE`. Additional parameters are allowed as well, which are passed on to the internal `dotplot()` function (`Sarkar, 2008`) and are used to format the resulting plot. In this case, the example syntax below also specifies the `xlab=` and `ylab=` parameters, labelling the two axes. The resulting plot is shown in Figure 2.3. These kinds of plots can be used to more easily assert the influence a grouped set of observations exert on the outcomes of analyses, relative to the influence excerted by other groups of observations.

In this case, it (again) is clear that the observation of the level of class structure of school number 7472 excerts the highest influence. This is based on

[3] Technically, the `ME.cook()` function internally forwards its output to the d`p.ME.cook()` function, which prepares the plotting and uses the lattice `dotplot()` function to create the actual plot.

Figure 2.3: Cook's Distances in Analysis of `school23` Data



the calculated value for Cook's distance, as well as that this influence clearly exceeds that of other schools.

```
ME.cook(estex.model, plot=TRUE,
    xlab="Cook's Distance, Class structure",
    ylab="School")
```

2.3.3 Exclude Influence, and Repeat

Based on the analyses and graphs shown in the previous sections, there are strong indications that the observations in school number 7472 excert too much influence on the outcomes of the analysis, and thereby unjustifiably determine

the outcomes of these analyses. To definitively decide whether or not the influence of these observations indeed is too large, the value of Cook's distance of this school can be compared with a cut-off value given. Regarding Cook's distance, it has been argued that observations exceeding a Cook's distance of $4/n$ are too influential Belsley et al. (1980), and need to be dealt with. In this formula, 'p' refers to the number of predictors on which Cook's distance was calculated. In the case of mixed effects models, this refers to the number of groups in which the observations are nested.

The Cook's distance of school number 7472 was determined to be 1.31, which readily exceeds the cut-off value of $(4/23) = .17$.[4] Thus, is can be concluded that the influence school number 7472 needs to be excluded form the analysis, before the results of that analyses are interpreted. This is done using the function `exclude.influence()`. This function basically has three parameters: first, the model from which the influence of some observations is to be excluded needs to be specified, together with the grouping factor and the specific level of that grouping factor in which the said observations are nested. The function modifies the original model and returns a new model, which can be checked again for possible influential data.

In the example below, the influence of school number 7472 is excluded from the orginal regression model, which was assigned to object '`model`' in section 2.1.

The result of the exc`lude.influence()` function again has the form of a mixed effects model and is here assigned to object `model.2` (again, this name is to chosen by the user).

```
model.2 <- exclude.influence(model, "school.ID", "7472")
summary(model.2)
```

Functions that work with 'normal' mixed effects models estimated with `lme4`, also work with models that were modified with the e`xclude.influence()` function. So, also a summary of `model.2` was requested, which is shown below. A few things are clear from this output. The estimate of the effect of class structure is now much stronger (-4.55) and statistically significant (t=2.95). This corresponds to what may have been expected based on the graphical representation of the data in Figure 2.1. Some other changes have been made to the model as well. The original intercept vector (which originally was indicated by `(Intercept)`) is now replaced by a variable called `intercept.alt`. This variable is basically an ordinary intercept vector (thus, with a value of 1 for each observation), except for the observations that are nested in the excluded nesting group. For these observations, the `intercept.alt` variable has score 0. Also, a new variable called `estex.7472` is shown. This variable is a dummy variable, indicating the observations that are nested in school number 7472. One such dummy variable is added to the model for each nesting group the influence of

[4] In this example p=2, for Cook's distance was calculated based on all parameters in the model, that is the intercept and the estimate for the structure variable. The number of groups (schools) observations are nested in is 23 in this example, thus n=23.

which is excluded. Generally, these modifications of the model ensure that the observations nested within the excluded nesting group do not contribute to the estimation of both the level and the variance of the intercept, and do not alter the higher level estimates unjustifiably.

```
Linear mixed model fit by REML
Formula: math ~ intercept.alt + estex.7472 + structure +
(0 + intercept.alt | school.ID) - 1
   Data: ..2
  AIC  BIC logLik deviance REMLdev
 3792 3814  -1891     3790    3782
Random effects:
 Groups    Name          Variance Std.Dev.
 school.ID intercept.alt 17.874   4.2277
 Residual                81.301   9.0167
Number of obs: 519, groups: school.ID, 23
Fixed effects:
              Estimate Std. Error t value
intercept.alt   69.346      6.314  10.983
estex.7472      54.839      3.617  15.163
structure       -4.550      1.545  -2.945
Correlation of Fixed Effects:
           intrc. e.7472
estex.7472  0.843
structure  -0.987 -0.854
```

As is shown in the procedural schematic in Figure 2.2, it is advisable to repeat this procedure to the point that the user is satisfied with the stability of the model, for instance when no group of observations exceeds the cut-off value. To do this in this example, the `model.2` object is again input to the `estex()` function, the results of which are stored in a second altered estimates object which we call `estex.model.2`:

```
estex.model.2 <- estex(model.2, "school.ID")
ME.cook(estex.model.2, plot=TRUE,
    xlab="Cook's Distance, Class structure",
    ylab="School",
    cutoff=.18)
```

Again, `ME.cook()` is used to calculate the values for Cook's distance, which returns the output shown below. School number 62821 is associated with the largest value for Cook's distance (.39). The cut-off value now differs (slightly) from the previous one, for the number of (effective) groups in which the observations are nested is decreased by 1, for the influence of school number 7472 was excluded. Thus, the cut-off value now is $(4/22) = .18$. Based on the output below, it can thus be concluded that school number 62821 is influential as well.

Finally, the call for `ME.cook()` in the syntax example above shows one more distinguishing characteristic. Again `plot=TRUE` is specified, together with specifications for labels on both the `x` and `y` axes. A plot of the Cook's distances is thus created, shown in Figure 2.4. In addition to this, the cut-off value of .18 is now indicated as well using `cutoff=.18`. As a result of this, all Cook's distances with a value larger than .18 will be indicated differently in the plot, as is the case in Figure 2.4 regarding the two schools numbered 62821 and 7474. Note that the Cook's distance for school number 7472 now equals 0, indeed, indicating that this school now no longer influences the parameter estimates.

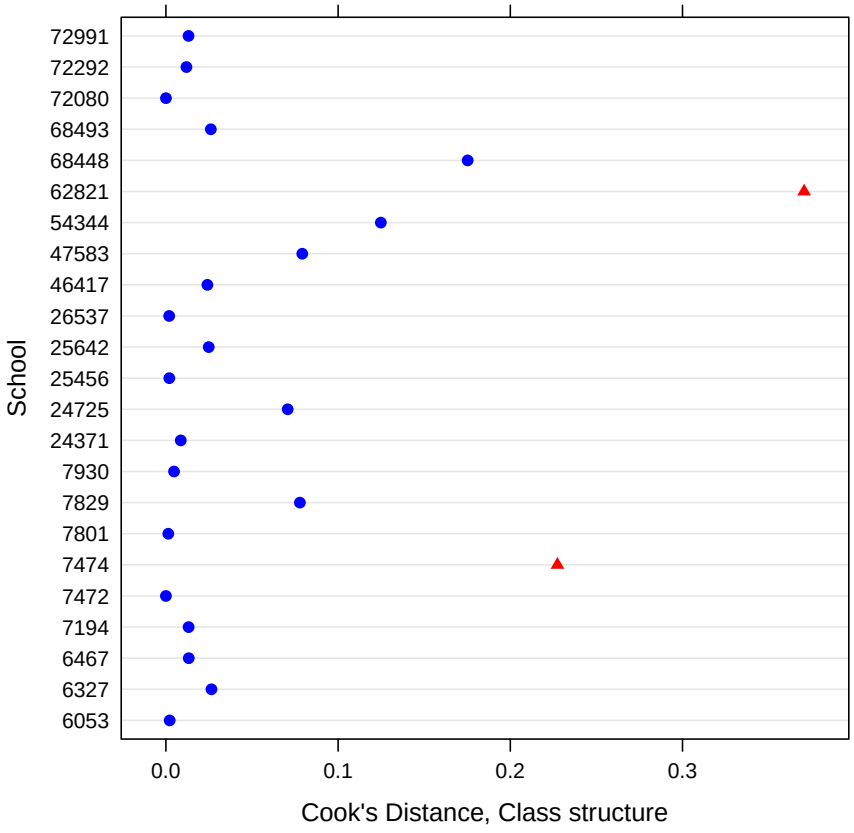
```
              [,1]
6053  2.186203e-03
6327  2.645659e-02
6467  1.326879e-02
7194  1.319258e-02
7472  0.000000e+00
7474  2.273674e-01
7801  1.378937e-03
7829  7.780663e-02
7930  4.728342e-03
24371 8.621802e-03
24725 7.072999e-02
25456 1.985731e-03
25642 2.487072e-02
26537 1.900817e-03
46417 2.409483e-02
47583 7.919332e-02
54344 1.248145e-01
62821 3.706191e-01
68448 1.752182e-01
68493 2.607158e-02
72080 2.669324e-05
72292 1.193296e-02
72991 1.311974e-02
```

Further analysis of this example would thus entail the exclusion of the influence of observations nested within school number 62821, and then to recheck the model by running through the three steps of the procedure again (Figure 2.2). This is not shown here, to not make this exercise overly lengthy.

2.4 Models with Multiple Variables

Up to this point, the model that has been evaluated was only made up of one single predictor variable `structure`. In practical applications, though, models often have more than a single predictor variable. The `influence.ME` package

Figure 2.4: Cook's Distances in (Modified) Analysis of school23 Data.



can deal with these kind of models without a problem, and has tools to compute appropriate measures of influence for these models.

For the examples in this section, a new regression model will be estimated using the `school23` data. In extension of the previous model, now also the students' social economic status (SES) and the school's mean level of SES are included. In addition to this, the students' time spend on their math homework is taken into account. In the syntax below, the `homework` variable is recoded from a factor variable, to a numerical representation of the amount of time spend on homework.

```
school23 <- within(school23, homework <- unclass(homework))
model.4 <- lmer(math ~  homework + structure + SES + mean.SES +
    (1 | school.ID), data=school23)
print(model.4, cor=FALSE)
```

Note that now `print(model.4, cor=FALSE)` is used to obtain a summary of the model. Although internally the `summary()` function is still used, this way the correlation matrix between the fixed effects is not printed, resulting in a more clear overview of the information required here. The summary of the `model.4` now is the following:

```
Linear mixed model fit by REML
Formula: math ~ homework + structure + SES + mean.SES +
    (1 | school.ID)
   Data: school23
  AIC  BIC logLik deviance REMLdev
 3692 3722  -1839     3684    3678
Random effects:
 Groups    Name        Variance Std.Dev.
 school.ID (Intercept) 11.146   3.3385
 Residual              66.992   8.1849
Number of obs: 519, groups: school.ID, 23
Fixed effects:
            Estimate Std. Error t value
(Intercept)  48.2986     4.4570  10.837
homework      2.1377     0.2700   7.917
structure    -0.7717     1.1096  -0.695
SES           3.3490     0.5793   5.781
mean.SES      2.2188     1.6288   1.362
```

Whereas Cook's distance provides an overview measure of influence of a set of observations on the parameter estimates of multiple variables, DFBETAS provides a standardized measure of change on each individual parameter estimate. This allows for a more elaborate evaluation of the changes to the model that were brought about by the influence excerted by the set of observations.

To calculate measures of DFBETAS, first the altered estimates of `model.4` are to be obtained, again using `estex()`. No different specification of `estex()` is required for use with the `ME.dfbetas()` function, as both the `ME.cook()` and the `ME.dfbetas()` functions work based on the information contained in the object returned by `estex()`.

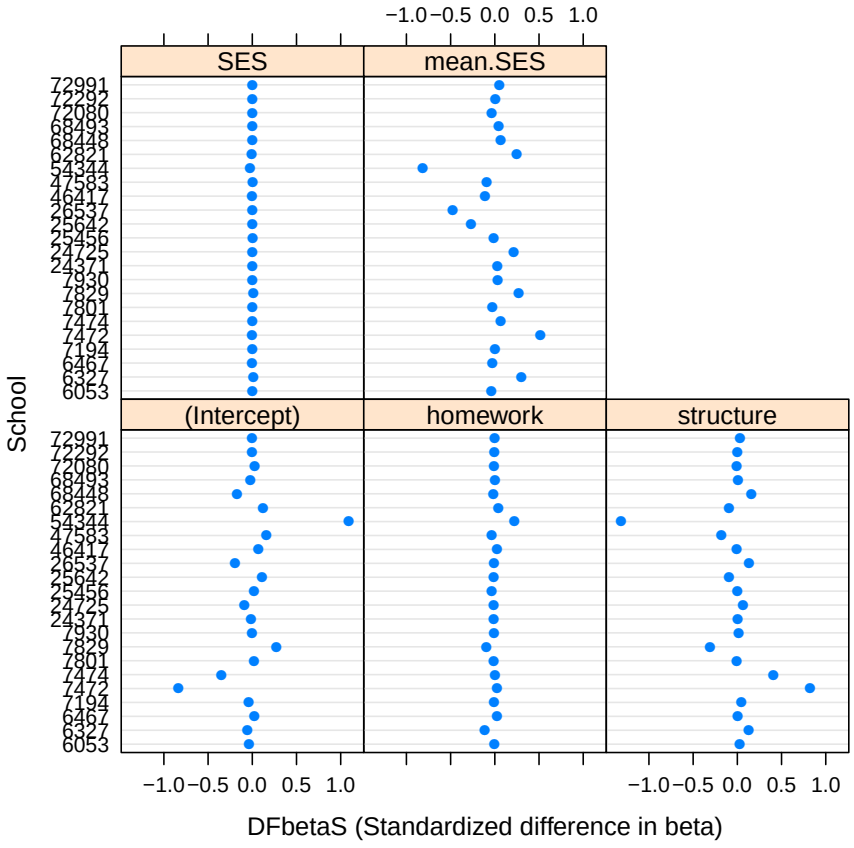
```
estex.model.4 <- estex(model.4, "school.ID")
ME.dfbetas(estex.model.4, plot=TRUE,
    xlab="DFbetaS (Standardized difference in beta)",
    ylab="School")
```

This results in a substantial amount of output, as well as a graph (Figure 2.5). The cut-off value of DFBETAS equals to $2/\sqrt{n}$ Belsley et al. (1980), which in this case is $2/\sqrt{23} = .41$. Based on the graphic, it is clear that, besides the intercept variable, the `mean.SES` and the `stucture` variables are susceptible to the influence of observations that are nested in single schools, whereas the `SES` and `homework` variables are not. This distinction is not all that surprising, since the `SES` and `homework` variables vary at the individual level, whereas `mean.SES` and `structure` are measured at the level of the school.

	(Intercept)	homework	structure	SES	mean.SES
6053	-0.0349846280	-0.0063492706	0.0277833746	7.421706e-04	-0.037169628
6327	-0.0551385767	-0.1124772319	0.1279844303	1.319926e-02	0.299357921
6467	0.0232880344	0.0274436711	0.0048958357	-3.205730e-03	-0.026542031
7194	-0.0382226925	-0.0088224934	0.0455556622	1.031353e-03	0.002230527
7472	-0.8363653273	0.0261386324	0.8245643171	-3.052015e-03	0.513982292
7474	-0.3492577843	0.0030813549	0.4086361155	-3.596427e-04	0.066523866
7801	0.0197877512	-0.0107712870	-0.0060458460	1.259011e-03	-0.027977363
7829	0.2735937368	-0.0947474315	-0.3065728205	1.108804e-02	0.271483748
7930	-0.0010442815	-0.0088979147	0.0154780476	1.040015e-03	0.033050124
24371	-0.0135529699	-0.0120436597	0.0052217940	1.408003e-03	0.030694778
24725	-0.0906477809	-0.0103982145	0.0642930855	1.214694e-03	0.214309471
25456	0.0218208860	-0.0358363215	0.0013631479	4.197131e-03	-0.012149346
25642	0.1111681053	-0.0108289432	-0.0925125201	1.265287e-03	-0.267657175
26537	-0.1956763561	-0.0065855014	0.1316211564	7.690705e-04	-0.475835993
46417	0.0690489529	0.0241727569	-0.0076382108	-2.818986e-03	-0.109907261
47583	0.1598235067	-0.0351731015	-0.1794321229	4.117460e-03	-0.091832269
54344	1.0923092155	0.2230415964	-1.3140860496	-2.584434e-02	-0.814782653
62821	0.1224894759	0.0423956141	-0.0934584810	-4.967902e-03	0.246802224
68448	-0.1716047053	-0.0175175480	0.1592892662	2.047680e-03	0.066998620
68493	-0.0225437153	0.0021687445	0.0098878259	-2.535315e-04	0.045688387
72080	0.0266997105	-0.0087803498	-0.0050217462	1.026084e-03	-0.035370530
72292	-0.0006333574	-0.0059408103	-0.0003039215	6.943779e-04	0.004972600
72991	-0.0008509988	0.0002349054	0.0295498482	-2.744565e-05	0.052888353

The numerical output given above by the `ME.dfbetas()` function provide a detailed report of the values of DFBETAS in the model. For each variable,

Figure 2.5: DFBETAS on model.4



as well as for each nesting group (in this example: each school), a value for DFBETAS is computed and reported upon. These numbers can be compared with the cut-off value all at once, by using the example below. Please note the use of the `abs()` function, which in this case returns the absolute values of DFBETAS.

```
abs(ME.dfbetas(estex.model.4)) >= .41
      (Intercept) homework structure   SES mean.SES
6053        FALSE    FALSE     FALSE FALSE    FALSE
6327        FALSE    FALSE     FALSE FALSE    FALSE
6467        FALSE    FALSE     FALSE FALSE    FALSE
7194        FALSE    FALSE     FALSE FALSE    FALSE
7472         TRUE    FALSE      TRUE FALSE     TRUE
7474        FALSE    FALSE     FALSE FALSE    FALSE
7801        FALSE    FALSE     FALSE FALSE    FALSE
7829        FALSE    FALSE     FALSE FALSE    FALSE
7930        FALSE    FALSE     FALSE FALSE    FALSE
24371       FALSE    FALSE     FALSE FALSE    FALSE
24725       FALSE    FALSE     FALSE FALSE    FALSE
25456       FALSE    FALSE     FALSE FALSE    FALSE
25642       FALSE    FALSE     FALSE FALSE    FALSE
26537       FALSE    FALSE     FALSE FALSE     TRUE
46417       FALSE    FALSE     FALSE FALSE    FALSE
47583       FALSE    FALSE     FALSE FALSE    FALSE
54344        TRUE    FALSE      TRUE FALSE     TRUE
62821       FALSE    FALSE     FALSE FALSE    FALSE
68448       FALSE    FALSE     FALSE FALSE    FALSE
68493       FALSE    FALSE     FALSE FALSE    FALSE
72080       FALSE    FALSE     FALSE FALSE    FALSE
72292       FALSE    FALSE     FALSE FALSE    FALSE
72991       FALSE    FALSE     FALSE FALSE    FALSE
```

Although in this model the model parameters again seem to have been over overly influenced by the observations associated with school number 7472, the parameters in this model seem to be influenced even stronger by observations in school number 54344. Not only does the `structure` variable show a value for DFBETAS that exceeds the cut-off value, this also holds for the `mean.SES` variable.

Since mostly the variables measured at the level of the nesting group tend to be heavily influenced by groups of observations in mixed effects models, it is often advisable to focus the analyses of influential data in mixed effects models on these variables. To accomodate for this, the `ME.cook()` function allows to specify a selection of variables on which the values for Cook's distance are to be calculated. Note, that whereas DFBETAS always relates to single variables, and thus does not change when calculated based on a different subset of variables, this is not the case for Cook's distance, which is a summary measure of changes

on all parameter estimates it is based on. Reports on Cook's distance thus should alway specify on which variables these values are based.

Cook's distance based on a subset of variables is calculated using `ME.cook()` with an extra specification of `parameters=...` . The `parameter` specification requires a single index-number of the parameter on which Cook's distance should be computed, or a vector of variable indeces, created with `c()`. These indices correspond to the order in which parameter estimates appear in the summary of a model, or to the order in which they appear in the output of `ME.dfbetas()` (which is identical). In this case, the variables of interest are `structure` and `mean.SES`, which are the third and fifth parameters in the model. The appropriate cut-off value with 23 nesting groups equals to $4/23 = .17$. Thus, to receive both numeric output and a graphical representation (Figure 2.6), the following specification of `ME.cook()` is given:

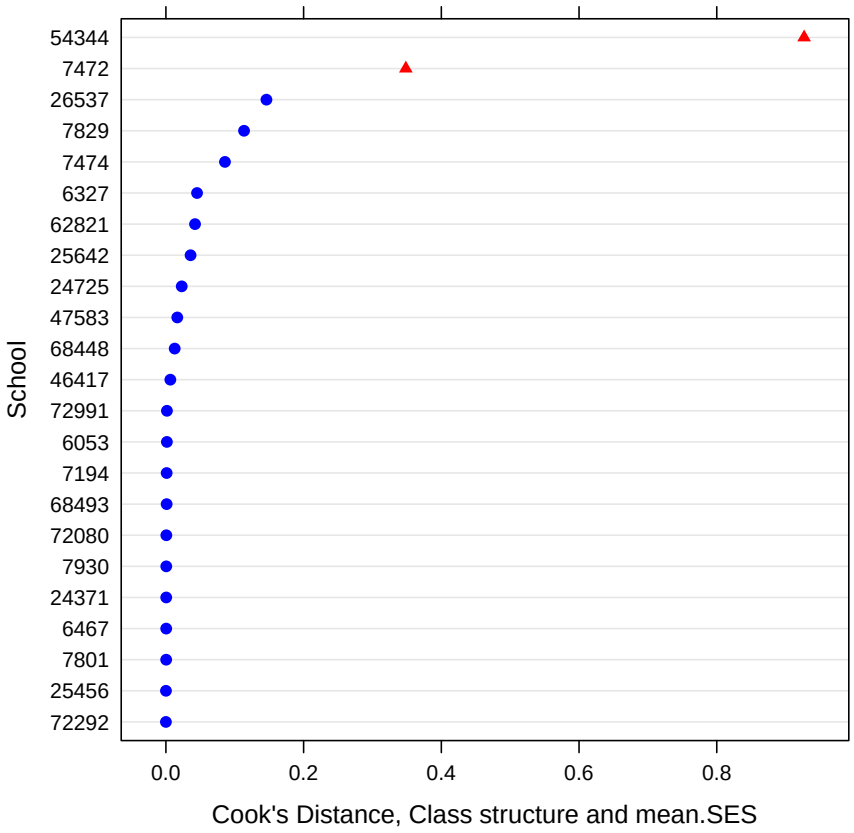
```
ME.cook(estex.model.4, parameters=c(3,5),
    plot=TRUE, cutoff=.17, sort=TRUE,
    xlab="Cook's Distance, Class structure and mean.SES",
    ylab="School")
```

The output below again shows one value of Cook's distance for each nesting group, in this case for each school. Both this table, as well as the graph show that the only school in which the combined measure of Cook's distance exceeds the cutoff value are the two school numbered 54334 and 7472. Another, final, characteristic of both the syntax above and the output below, is the use of the `sort=TRUE` parameter. Both the resulting numeric output and its graphical represention are now ordered. This is can be used, especially the graphical representation, to detect gaps in the magnitude of the Cook's distances, and can be used with the `ME.dfbetas()` function as well (but: refer to chapter 3 for details on how to use this option with `ME.dfbetas()`).

The user can now decide how to procede, following the scheme shown in Figure 2.2.

```
              [,1]
72292 1.625142e-05
24371 8.197900e-04
7194  9.746399e-04
6053  1.137111e-03
7930  1.222035e-03
7801  1.455177e-03
68493 1.832051e-03
25456 2.994084e-03
72080 3.282086e-03
72991 4.507696e-03
6467  4.791353e-03
68448 1.033409e-02
47583 1.205566e-02
24725 2.184971e-02
```

Figure 2.6: Cook's Distances based on structure and mean.SES variables




```
46417 2.595337e-02
25642 2.821170e-02
62821 2.958245e-02
6327  4.506893e-02
7474  7.015919e-02
7829  8.242500e-02
26537 1.042892e-01
7472  2.409327e-01
54344 7.188965e-01
```

Chapter 3

A Detailed Account of influence.ME

3.1 Getting Started With R

The R software is developed by the R Core Development Team, presently having nineteen members (R Development Core Team, 2008). It has a syntax-driven interface which allows for a high level of control, many add-on packages, an active community supporting the program and it's users, and an open structure. All in all, it aims to be statistical software that goes beyond pre-set analyses. The source code of the R software and its contributed packages are distributed with an open-source license. This means that everybody is allowed to read and change the program code. The consequence of this is that many people have written extensions to R which are able to nest itself in the fundaments of the software. For instance, it can interact with programs such as (WIN)BUGS or have extensions based on R code, as well as other extensions based on other programming languages. `Influence.ME` is such an extension package.

A typical R session is characterized by its flexibility. The software is set up in such a way, that functions or command can interact and thereby be combined to new ones. Obviously many statistical methods are already available, but if a command just doesn't do exactly what you want it to do, it can easily be altered. Or, you build your analyses from the ground up using the most basic of functions. If you can think of it, you can create it.

3.1.1 Getting Help on R

Concordant with the open source community, R-Project is accompanied by many additional help functions. Most of them are freely available.

The help()-function R-Project has a built-in help function. This functionality is focused on informing the user on the parameters function have. Almost for all functions some examples are given as well. A general help

page is available, which contains several introductory documents like 'An Introduction to R', 'Frequently Asked Questions', and 'The R Language Definition'. More advanced documents are made available as well, such as 'Writing R Extensions' and 'R Internals'. This general help page is called for by entering: help.start(). To obtain help on a specific function, you use `help()` with the name of the function between the brackets. For instance, if you want help on the `plot` function, use the following syntax: `help(plot)`. This results in a page that gives a short definition of the function, shows the parameters of the function, links to related functions, and finally gives some examples.

Freely available documents More elaborate documents can be found on the website of R-Project (http://www.r-project.org) in the documents section. This can be found by clicking on 'manuals' from the home-page, just below the 'documents' header. First, a couple of documents written by the core development team of R-Project are offered. Second, the 'Contributed Documentation' leads to many more documents.

Books on R-Project Many books have been written on R-Project, ranging from very basic-level introductions to the ones that address the fundamental parts of the software. Some excellent introductions include:

- An R and S-PLUS Companion to Applied Regression, by John Fox (2002)
- Multilevel / Hierarchical Models, by Andrew Gelman and Jennifer Hill Gelman and Hill (2007)

R-help mailinglist When all help fails, there is always the R-Help mailing-list. This is a service where all members receive the e-mails that are send to a specific address. The quality and speed of the given answers and solutions is often very high. Questions are asked and answered many times a day, so be prepared to receive a high volume of e-mail when signing up for this service. More information on the R-help mailing-list, as well as the ability to sign-up, can be found on: https://stat.ethz.ch/mailman/listinfo/r-help

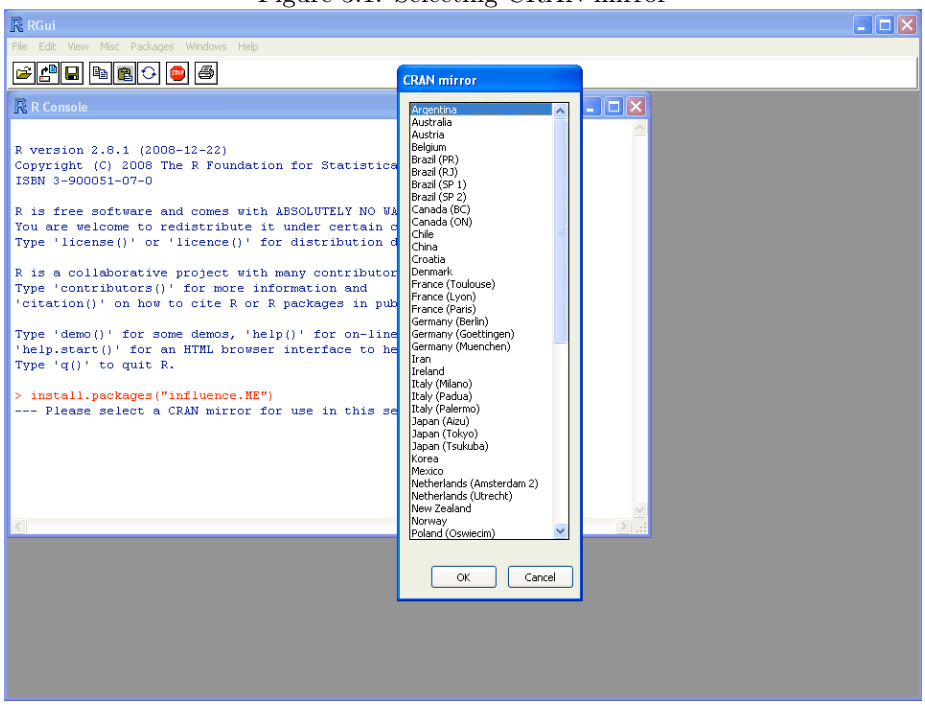
3.1.2 Installing Packages

A freshly installed version of R-Project can do some pretty nice things already, but much more functionality can be obtained by installing packages that contain new functions. To start using influence.ME, this package should be installed first. These packages are available on a central archive, called CRAN (Comprehesive R Archive Network), and can easily be installed from within R-Project. CRAN is mirrored throughout many places in the world, so there is always a repository close, and available.

To install the package influence.ME, simply type the following syntax at the R prompt:

```
install.packages(''influence.ME'')
```

Figure 3.1: Selecting CRAN mirror



This automatically downloads the influence.ME package from a CRAN mirror, and installs it. In some cases, if no default CRAN mirror ('server') has been set, the user is prompted to choose one of the available mirrors. In Windows XP, this looks like Figure 3.1. In principle, each mirror contains exactly the same information and packages. Users can select one to their likings, based on the criterium of vicinity, or previous experiences. Should a mirror (temporarily) cause problems, select another one by clicking on 'Packages' –> 'Select CRAN mirror'... in the R GUI.

3.2 Basic Rationale of Determining Influential Data

The basic rationale behind influential data is that the estimates of statistical models, aimed at making inferences, should not be dominated by single observations. In other words, observations are regarded to be too influential, when large differences in model estimates arise when these observations are excluded from the model. Observations can be influential based on a combination of being an outlier (large difference between observed and predicted scores) and high leverage (ie. extremely high or low scores on the x-variables). Although related, being an outlier and having high leverage do not always coincide: due to high leverage an observation can influence model parameters so strongly, that the

regression slope is drawn towards this observation which, as a result, does no longer appear as an outlier.

Mixed effects regression models tend to become common practice in the field of social sciences, but diagnostic tools to evaluate these models lag behind. It is however commonly accepted that tests for influential cases should be performed, especially when estimates are based on a relatively small number of cases. Testing for influence with mixed effects models is especially important in Social Science applications, for two reasons. First, models in the social sciences are frequently based on large numbers of individuals while the number of higher level units is often relatively small. Secondly, often the higher level units are remarkably similar, for instance in the case of neighboring countries.

To apply the same logic to mixed effects models one has to measure the influence of a particular higher level unit on the estimates of a higher level predictor. One approach to do this was suggested by Langford and Lewis (1998). In this approach, the mixed effects model is adjusted by *neutralizing* the unit's influence on that estimate, while at the same time allowing the unit's lower-level cases to help estimate the effects of the lower-level predictors in the model. This procedure is based on a modification of the intercept and the addition of a dummy variable for the cases that might be influential. Secondly, a more classical approach for detecting influential data, entails the iteratively *deletion* of the higher level unit and its' underlying lower-level units (Snijders and Berkhof, 2008; Snijders and Bosker, 1999). This method is available in `influence.ME` as well. Regarding mixed models, this entails the deletion of all observations nested within a higher level unit. In the next section, these procedures are decribed in more detail.

3.3 Internal Modification of the Data

As discussed in the previous section, `influence.ME` provides two ways in which the influence of variables measured at the higher-level can be determined, in order to evaluate the extend to which these higher-level units (overly) influence the outcomes of the model. These two methods are described in this section.

3.3.1 Modification Intercept + Dummy

The default method in `influence.ME` to neutralize the influence of the higher-level unit, while still allowing the unit's lower-level cases to help estimate the effects of the lower-level predictors in the model. For each higher-level unit evaluated based on this method, the intercept-vector of the model is set to 0, and an (additional) dummy variable is added to the model, with score 1 for the respective higher level unit. If multiple higher-level units are evaluated simultaneously, the intercept-vector is set to 0 for all these units, and one dummy variable is added for each.

To illustrate this internal modification of the data, again the simple model from chapter 2 is estimated below.

```
data(school23)
model <- lmer(math ~  structure + (1 | school.ID),
    data=school23)
model.frame(model)[40:49,]
```

The data that is used to estimate the mixed model is stored inside the model object, and can be extracted using the `model.frame()` function. Using the last row of the syntax above, the 40th to 49th row of this model.frame are shown:

```
> model.frame(model)[40:49,]
   math structure school.ID
40   48         3      6053
41   65         3      6053
42   41         3      6053
43   50         3      6053
44   64         3      6053
45   61         4      6327
46   56         4      6327
47   46         4      6327
48   60         4      6327
49   65         4      6327
```

The model.frame of this simple model contains three variables: the dependent variable `math`, the predictor variable `structure`, and the grouping factor `school.ID`. Note that no intercept is present: normally this is internally added by the `lme4` functions but not stored in the model.frame because it was not explicitly present in the original data. These ten rows shown above shown data from 2 different schools: school 6053 and school numer 6327.

```
model.2 <- exclude.influence(model, "school.ID", "6053")
fixef(model.2)
model.frame(model.2)[40:49,]
```

In the syntax above, the exclude.influence() function is used to exclude the influence exerted by the first of these schools (6053) from the estimtates of 'model'. Just as in the example of chapter 2, the resulting model is stored in an object, which is called `model.2` here. Instead of a full summary, `fixef()` is used to only have the estimates of the fixed effects returned. And, again, the same ten rows of the modified model.frame are shown.

```
> fixef(model.2)
intercept.alt    estex.6053     structure
    58.642630     62.449717     -2.043845
> model.frame(model.2)[40:49,]
   math intercept.alt estex.6053 structure school.ID
40   48             0          1         3      6053
41   65             0          1         3      6053
```

```
42   41             0           1          3      6053
43   50             0           1          3      6053
44   64             0           1          3      6053
45   61             1           0          4      6327
46   56             1           0          4      6327
47   46             1           0          4      6327
48   60             1           0          4      6327
49   65             1           0          4      6327
```

This shows a few deviations from the previous example: two columns are added to the model.frame: one called `intercept.alt`, and one called `estex.6053`. The intercept.alt variable is a vector of 1's, and replaces the 'internal' intercept of the `lme4` functions, which is usually indicated with `(Intercept)`. The procedure of neutralizing the influence exerted by one or more grouping factors entails modifying the intercept in such a way, that it has a value 0 for these nesting groups, and a separate dummy variable is added to the model for each of these groups. These added dummies are all called '`estex.`' with the addition of the value of the specific grouping level.

Indeed, in the output above, both the `intercept.alt` and the `estex.6053` variable are present in the fixed estimates. From the extract of the model.frame it is clear, that the modified intercept variable (`intercept.alt`) indeed has value 0 for students in school 6053, just as the dummy variable has value 1 for these students. In this particular example, the `intercept.alt` and `estex.6053` variables are opposites.

```
model.3 <- exclude.influence(model.2, "school.ID", "6327")
fixef(model.3)
model.frame(model.3)[40:49,]
```

In the final example, for which the syntax is shown above, the influence of a second nesting group will be excluded from the model. Again, `exclude.influence()` is used, with `model.2` as input model. The resulting fixed effect estimates (below) now show that an additional dummy variable was added to the model, called `estex.6327`. Also, the model.frame has been modified again: now the `intercept.alt` variable has a value 0 for students in both schools 6053 and 6327. The dummy variables `estex.6327` and `estex.6053` only indicate a single nesting group each.

```
> fixef(model.3)
intercept.alt    estex.6327    estex.6053     structure
    58.386099     65.567457     62.462524     -2.048114
> model.frame(model.3)[40:49,]
   math intercept.alt estex.6327 estex.6053 structure school.ID
40   48             0          0          1         3      6053
41   65             0          0          1         3      6053
42   41             0          0          1         3      6053
```

```
43   50          0          0          1          3       6053
44   64          0          0          1          3       6053
45   61          0          1          0          4       6327
46   56          0          1          0          4       6327
47   46          0          1          0          4       6327
48   60          0          1          0          4       6327
49   65          0          1          0          4       6327
```

3.3.2 Deletion of Observations

In addition to the advanced method described above, `influence.ME` also provides means for excluding the influence of a higher level unit by simply deleting the higher level unit, and with it the lower level units nested within. Since this functionality is not the default option in `influence.ME`, it must be specifically specified, by adding the `delete=TRUE` parameter to the `estex()` or `exclude.influence()` functions. To illustrate this internal modification of the data, the same simple model as in the previous section is used:

```
data(school23)
model <- lmer(math ~  structure + (1 | school.ID), data=school23)
model.frame(model)[40:49,]
```

The data that is used to estimate the mixed model stored inside the model object, and can be extracted using the `model.frame()` function. Using the last row of the syntax above, the 40th to 49th row of this model.frame are shown:

```
> model.frame(model)[40:49,]
   math structure school.ID
40   48         3      6053
41   65         3      6053
42   41         3      6053
43   50         3      6053
44   64         3      6053
45   61         4      6327
46   56         4      6327
47   46         4      6327
48   60         4      6327
49   65         4      6327
```

To allow a comparison with the default procedure for excluding the influence of a higher level unit, the same school is excluded as in the previous section. However, now school number 6053 and all observations nested within are actually deleted from the data, using the `delete=TRUE` parameter:

```
model.2 <- exclude.influence(model, "school.ID", "6053", delete=TRUE)
fixef(model.2)
model.frame(model.2)[40:49,]
```

In the syntax above, the `exclude.influence()` function is used to exclude the influence exerted by the first of these schools (6053) from the estimtates of 'model'. Just as in the example in section 3.3.1, the resulting model is stored in an object, which is called `model.2` here. Instead of a full summary, `fixef()` is used to only have the estimates of the fixed effects returned. And, again, the same ten rows of the modified model.frame are shown.

```
> fixef(model.2)
(Intercept)   structure
  58.641314   -2.043401
>
> model.frame(model.2)[40:49,]
   math structure school.ID
84   53         2      7472
85   42         2      7472
86   43         2      7472
87   57         2      7472
88   33         2      7472
89   64         2      7472
90   36         2      7472
91   56         2      7472
92   48         2      7472
93   48         2      7472
>
```

What is clear from the fixed parameters in the output above, is that indeed no modification of the intercept, or addition of a dummy, took place. This is also shown by the selected rows in the model.frame. In addition, it is shown there that the original observations nested in school number 6053 now are absent (the row numbers in the left-most column still reflect the row numbers in the original data, before rows were deleted).

3.3.3 A Comparison of Both Methods

Although no thorough study has (yet) been made to the differences in outcomes derived from applying both methods, no large differences are expected. Of course, users can evaluate differences between the two methods for their own models. In this section, a strategy for doing so is suggested, based on a simple model of the school23 data. For this purpose, the model with multiple variables from section 2.4 will again be estimated.

```
school23 <- within(school23, homework <- unclass(homework))
model.4 <- lmer(math ~  homework + structure + SES + mean.SES +
    (1 | school.ID), data=school23)
```

Now, the first step in the procedure for `influence.ME` (see Figure 2.2) will be taken twice. First, the influence of higher level units is excluded based on the

default method, entailing a modification of the intercept and the addition of a dummy variable. Second, the higher level units and its underlying observations are deleted alltogether.

```
estex.model.mod <- estex(model.4, "school.ID")
estex.model.del <- estex(model.4, "school.ID", delete=TRUE)
cook.mod <- ME.cook(estex.model.mod, parameters=c(3,5))
cook.del <- ME.cook(estex.model.del, parameters=c(3,5))
```

Secondly, in the syntax above, the Cook's distances are calculated twice. The model estimates from which iteratively the influence of higher level units was excluded using the default method (modification intercept + dummy) were assigned to the object `estex.model.mod`, and the Cook's distances based on these were assigned to an object named `cook.del`. The model estimates from which iteratively the influence of higher level units was excluded by deleting the respective observations, were assigned to the object `estex.model.del`, cook's distances based on which are stored in the object `cook.del`.

```
xyplot(cook.del ~ cook.mod,
    type=c("p", "g"),
    ylab="Cook's Distance (deleted observations)",
    xlab="Cook's Distance (modification intercept + dummy)")
```

The resulting cook's distances, calculated using different methods, are plotted against each other using the syntax above. The resulting plot is shown in Figure 3.2. Although minor differences exist, all points are near the diagonal of the plot. Some cook's distances based calculated based on models from which observations were deleted seem to be slightly higher. Nevertheless, conclusions drawn based on both analyses lead towards exactly the same conclusion.

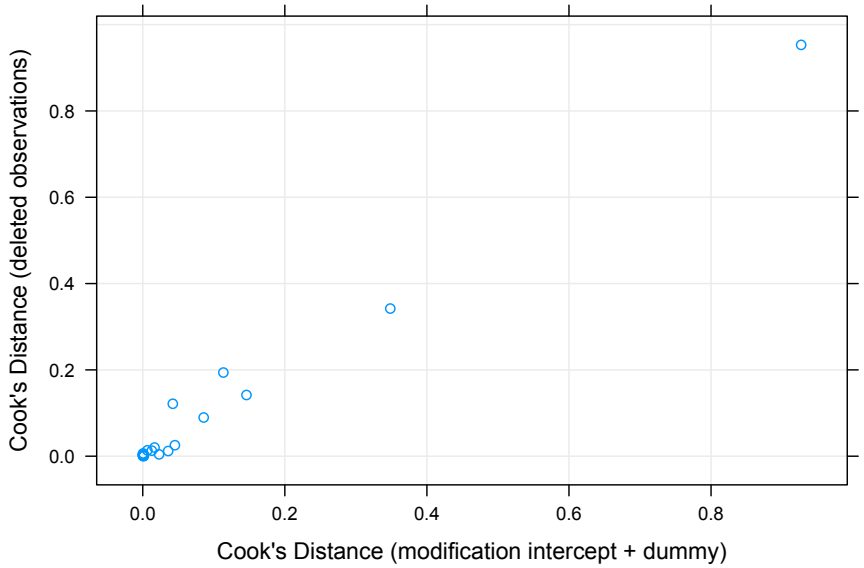
3.4 The Outcome Measures

3.4.1 DFBETAS

DFBETAS is a standardized measure of influence that indicates the level of influence observations have on single parameter estimates (Fox, 2002). Regarding mixed models, this relates to the influence a higher-level unit (and observations nested within it) has on the parameter estimate, and is calculated based on the absolute difference in the magnitude of the parameter estimate between the models both including, and excluding, (the influence of) the higher level unit. This absolute difference is divided by the standard error of the parameter estimate *excluding* the observations nested in the higher level unit under investigation.

$$dfbetas_{ij} = \frac{\hat{\gamma}_i - \hat{\gamma_{i(-j)}}}{se(\hat{\gamma_{i(-j)}})}$$

Figure 3.2: Comparison of Cook's distances calculated using different methods



As a rule of thumb, a cut-off value is given for DFBETAS (Belsley et al., 1980):

$$2/\sqrt{n}$$

in which n, the number of observations, refers to the number of groups in the grouping factor under evaluation (and not, for instance, to the number of observations nested in the group under investigation).

3.4.2 Cook's Distance

Since DFBETAS provides a value for each parameter and for each higher-level unit that is evaluated, this often results in quite a large number of values to evaluate (Fox, 2002). An alternative is provided by Cook's distance, a highly commonly used measure of influence. Cook's distance provides a summary measure for the influence a higher level unit exerts on *all* parameter estimates simultaneously, or a selection thereof. A formula for Cook Distance is provided (Snijders and Bosker, 1999; Snijders and Berkhof, 2008):

$$C_j^{0F} = \frac{1}{r+1}(\hat{\gamma} - \gamma_{(\hat{-j})})' \hat{\sum}_F^{-1} (\hat{\gamma} - \hat{\gamma}_{(-j)})$$

In which $\hat{\gamma}$ represents the vector of original parameter estimates, $\hat{\gamma}_{(-j)}$the parameter estimates of the model excluding higher-level unit j, and $\hat{\sum}_F$ rep-

resents the covariance matrix. In influence.ME, the covariance matrix of the model *excluding* the higher-level unit under investifation j is used. Finally, r is the number of parameters that are evaluated. This excludes the intercept vector, so if one evaluates a model with 3 variables associated with higher-level units, and one wants to include the intercept in the calcluation of Cook's distance, 1/r+1 = 1/4 is used. When the intercept is not included in the calculation of Cook's distance, 1/r = 1/3 is used.

Again, as a rule of thumb, a cut-off value is given for Cook's Distance (Van der Meer et al., 2009):

$$\frac{4}{n}$$

in which p is the number of parameters on which the value for Cook's distance was used (plus 1 for the intercept if it was included in the calculation of the intercept) and n the number of observations, refers to the number of groups in the grouping factor under evaluation.

> As a final note, it is pointed out that if Cook's distance is calculated based on a single parameter, the Cook's distance equals the squared value of DFBETAS for that parameter. Naturally, this is reflected in the cut-off values as well:
>
> $$\sqrt{\frac{4}{n}} = \frac{2}{\sqrt{n}}$$

3.4.3 Percentile Change

Depending upon the goal for which the mixed model is estimated (prediction vs. hypothesis testing), the use of formal measures of influence as DBBETAS and Cook's distance may be less desirable. The reason for this is that based on these measures it is not alway immediately clear to what extend the magnitudes of the parameter estimates change. A simple alternative is offered by the function `ME.pchange()`. For each higher-level unit the influence of which is iteratively excluded from the model, this function returns the change in the parameter estimates as a percentage of the complete model (including the higher-level unit under observation). This percentage is calculated based on the absolute difference between the parameter magnitude of the model both including and excluding the higher-level unit, divided by the parameter estimate of the complete model and multiplied by 100%. A percentage of change is returned for each parameter separately, for each of the higher-level units under investigation. In the form of a formula:

$$(\hat{\gamma} - \hat{\gamma_{(-j)}})\frac{1}{\hat{\gamma}} * 100\%$$

No cut-off value is provided, for determining what procentual change in parameter magnitude is considered too large, will primarily depend on the goal for which the model was estimated and, more specifically, the nature of the hypotheses that are tested.

3.5 Limits to Compatible Types of Mixed Effects Models

Attempting to remain true to the R design philosophy, the `influence.ME` package has been designed to interact with existing functions as closely as possible. In its present version, it works with models estimated using the `lme4` package Bates et al. (2008), which encompasses a wide variety of linear, generalized linear, and non-linear mixed models, with flexible nesting options. Models that have been modified by the `influence.ME` package are of the same class as models fitted using the `lme4` package. Existing helper functions that work with lme4-models, such as `summary()`, `print()`, and `update()`, therefore also work with models that have been manipulated using `influence.ME`.

As far as the authors are aware, `influence.ME` works with any mixed model estimated using the `lme4` package[1], except for a few conditions. This section focuses on these situations, and offers solutions for some.

3.5.1 Beware of Non-Converged Models

Unfortunately, not all models are equally unique. That is to say, that the optimization procedures estimating the model parameters, can run into problems due to various causes, such as poor model specification, weak data, multimodel likelihoods, and various others. Because the procedure for determining influential data, as it is incorporated in `influence.ME` entails the re-estimation of a mixed model with iteratively slight modifications made to it, the risk of running into non-converged models is present, even when the original model converged.

In some cases of poor model convergence, the estimation functions in the `lme4` package (`lmer`, `glmer`, and `nlmer`) detect problems and return warnings in addition to the possibly poorly converged model. This is, for instance, the case when the iteration limit has been reached, before optimalization occurred. Another example (not replicable based on the examples in this manual) is shown below:

```
estex.model <- estex(model, "grouping.ID")
Warning message: In mer_finalize(ans) : singular convergence (7)
```

R distinguishes between errors and warnings. In the case of an error, the internal loop of the `estex()` function breaks, and no output is returned other than an error message. Warnings, however, allow the function to continue, and are printed after the `estex()` function returned its output. Users should treat measures of influential data with care, when one or more warnings regarding the convergence are returned. If the models associated with one or more grouping levels converged poorly, the measures of influential data based on these models often unjustifiably indicate influential data. In this situation, the parameters of

[1] The authors, however, have as of yet not tested the `influence.ME` package using non-linear (`nlmer`) models. Therefore, the package is officially only supported for linear and generalized linear models.

these poorly converged models, on which the measures on influential data are based, may have substantially changed compared with a reference model, either because the associated grouping levels indeed are overly influential, or due to the poor convergence.

3.5.2 Factor Variables as Predictors

Unlike some other statistical packages, in R categorical (predictor) variables, when defined as factor variables, can be directly specified in model formulae. Based on various settings for contrasts, the variables are automatically recoded internally. However, in tis current state of development, `influence.ME` is not always compatible with models based on factor variables used as predictor variables (the grouping factor can be a factor variable without problems). More specifically, the method of excluding the influence of higher level units that incorporates the modification of the intercept with the addition of the dummy, cannot be used in coalition with a mixed model with one or more factor variables. This applies both to the `estex()` and the `exclude.influence()` functions.

Although future development of `influence.ME` will certainly focus on solving this problem, there are two solutions to this problem. The first is to manually recode the factor variables to dummy variables. These can be added to the model, leading to identical results, and then used with `influence.ME`.

```
table(school23$sex)
school23$female <- ifelse(school23$sex=="Female", 1, 0)
table(school23$female)
```

A simple example of recoding the sex variable to a dummy variable indicating women, is given above. Of course, using a table, the variable is investigated first. Below, it is shown that 249 male and 270 female students are represented in the school23 data.frame. Next, a dummy variable indicating the female students is created, using the `ifselse()` function. `ifelse()` checks a condition (in this case, whether a respondent's sex equals `'Female'`, and returns a value for success (1) and failure (0) of this test. These returned values are assigned to a variable called '`female'`, which is created inside the school23 data.frame. Again, this variable is tabulated, to check whether the coding is successful, which it is.

```
> table(school23$sex)
  Male Female
   249    270
> school23$female <- ifelse(school23$sex=="Female", 1, 0)
> table(school23$female)
  0   1
249 270
```

A second solution to this problem is to use the `delete=TRUE` parameter in either the `estex()` or the `exclude.influence()` funtions. As discussed in section 3.3.2, this is the classic method for determining influential data, and in a mixed

model setting this entails the deletion of observations nested within groups from the data.

A simple example is given below:

```
model <- lmer(math ~ sex + structure +
    (1 | school.ID), data=school23)
estex.model.mod <- estex(model, "school.ID")
estex.model.del <- estex(model,
   "school.ID", delete=TRUE)
model.7472.mod <- exclude.influence(model,
    "school.ID", "7472")
model.7472.del <- exclude.influence(model,
    "school.ID", "7427", delete=TRUE)
```

A straightforward model is estimated incorporating the sex of a student, and the level of class structure (this model is assigned to the object `model`). Sex is a factor variable, and thereby incompatible with the default option in the `estex()` and `exclude.influence()` functions. In the output below, the outcomes of the syntax above are shown. When the default option is used, retreiving the estimated from which iteratively the influence of the nesting groups was excluded with `estex()`, fails and returns an error message. When `delete=TRUE` is specified, all goes well. In the last two example, the influence of school number 7472 is excluded from the models. Again, due to the presence of a factor variable in the model, this only works when `delete=TRUE` is specified in the `exclude.influence()` function.

```
> estex.model.mod <- estex(model, "school.ID")
Error in mer_finalize(ans) :
    Downdated X'X is not positive definite, 4.
> estex.model.mod <- estex(model,
    "school.ID", delete=TRUE)
> model.7472.mod <- exclude.influence(model,
    "school.ID", "7472")
Error in mer_finalize(ans) :
    Downdated X'X is not positive definite, 4.
> model.7472.del <- exclude.influence(model,
    "school.ID", "7427", delete=TRUE)
```

Chapter 4

The Help Pages

4.1 The Core Functions

4.1.1 estex()

This function is the workhorse function of the influence.ME package. Based on a priorly estimated mixed effects regression model (estimated using lme4), the estex() function iteratively modifies the mixed effects model to neutralize the effect a grouped set of data has on the parameters, and which returns returns the fixed parameters of these iteratively modified models. These are used to compute measures of influential data.

Usage and Arguments

```
estex(model, group, select = 0, count = FALSE, delete=FALSE, ...)
```

model Mixed effects model of class 'mer' group.

Grouping factor in model of which iteratively levels are omitted

select Defines the selection of grouping factors that should be omitted. Defaults to 0, resulting in each level of the grouping factor being omitted iteratively. When a selection is defined, model parameters for the full model, and the altered model are returned. The selection can be a vector of multiple levels of the grouping factor.

count If count=TRUE, the remaining number of grouping factors that still need to be omitted are printed.

delete If delete=FALSE (default), the influence of higher level groups is excluded from the model by setting the intercept-vector for the observations nested within these groups to 0, and by adding a dummy-variable indicating these observations (Langford and Lewis, 1998). If delete=TRUE, the

influence is excluded by simply deleting the observations nested within the higher level group.

... Optional arguments that are passed on to the lmer/glmer function

Details

The basic rationale behind measuring influential cases is that when iteratively single units are omitted from the data, models based on these data should not produce substantially different estimates. To apply this logic to mixed effects models one has to measure the influence of a particular higher level unit on the estimates of a higher level predictor. This means that the mixed effects model has to be adjusted to neutralize the unit's influence on that estimate, while at the same time allowing the unit's lower-level cases to help estimate the effects of the lower-level predictors in the model. This procedure is based on a modification of the intercept and the addition of a dummy variable for the cases that might be influential.

estex() is the workhorse function of this likewise called package. Based on a priorly estimated mixed effects regression model (of the 'mer' class), the estex() function iteratively modifies the mixed effects model by neutralizing the effect a grouped set of data has on the parameters, and which returns returns the fixed parameters of these iteratively modified models.

The returned object (see 'Returned Value') contains information which is required for functions computing various measures of influential data.

Returned Value

The object returned by estex() of class "alt.est" contains the 'altered estimates' required by several other functions to calculate measures of influential data. A list containing six elements is returned:

or.fixed Fixed estimates of the original model (based on the full data)

or.se Standard Error of the estimates of the original model

or.vcov Variance / Covariance matrix of the original model

alt.fixed Matrix of the fixed parameters estimate, after iteratively subsets of data are removed. Altered estimates associated with the deletion of data nested within each grouping factor are provided.

alt.se Matrix of the standard errors of the fixed parameter estimates, after iteratively subsets of data are removed. Altered estimates associated with the deletion of data nested within each grouping factor are provided.

alt.vcov Variance / Covariance matrix of the altered models, after iteratively subsets of data are removed. Altered estimates associated with the deletion of data nested within each grouping factor are provided.

Note

Please note that in its present form, the estex function only works on mixed effects regression models of class mer that have been estimated using the functions in the lme4 package.

Also, it is required that the mer model was estimated using a factor variable to indicate group levels. When using something similar to + (1 | as.factor(variable)), the function is not able of identifying the correct grouping factors, and returns an error.

Since estex() entails the re-estimation of the provided mixed effects model for each level of the specified grouping factor (after alteration of the data), executing this procedure can be computationally highly demanding.

To facilitate the use of estex() with more complex models (i.e. models based on large amounts of data and / or with a large number of groups.

Examples

```
model <- lmer(diameter ~ (1|plate) + (1|sample), Penicillin)
alt.est <- estex(model, "plate")
alt.est
alt.est <- estex(model, "sample")
alt.est
```

4.1.2 exclude.influence()

Using mixed effects regression models, exclude.influence excludes the influence of a group of cases grouped within a single grouping factor, or a set of grouping factors. The function returns a model in which the influence a grouped set of observations has on both the variance and point-estimate of the (random) intercept.

Usage and Arguments

```
exclude.influence(model, grouping, level, delete=FALSE)
```

model A mixed effects regression model

grouping The grouping factor of which one or more groupings levels are to be 'neutralized'

level Vector of character strings, indicating either a single level or a set of grouping levels the influence of which is to be neutralized

delete If delete=FALSE (default), the influence of higher level groups is excluded from the model by setting the intercept-vector for the observations nested within these groups to 0, and by adding a dummy-variable indicating these observations (Langford and Lewis, 1998). If delete=TRUE, the

influence is excluded by simply deleting the observations nested within the higher level group.

Details

To apply the basic logic of influential cases to mixed effects models one has to measure the influence of a particular higher level unit on the estimates of a higher level predictor. This means that the mixed effects model has to be adjusted to neutralize the unit's influence on that estimate, while at the same time allowing the unit's lower-level cases to help estimate the effects of the lower-level predictors in the model. This procedure is based on a modification of the intercept and the addition of a dummy variable for the cases that might be influential.

The model that is returned by exclude.influence thus contains a modified intercept, and one or more additional dummy variables. To help identify this model as modified (which is required when in a later stage the influence of additional grouping levels is excluded), the intercept is renamed to 'intercept.alt'. The additional dummy variables, indicating the observations associated with the grouping factor levels of which the influence was neutralized, are labeled starting with 'estex.', combined with the label of the neutralized grouping level.

Returned Value

Mixed effects regression model of class 'mer', with a modified random intercept and dummy variables indicating the estimates of the neutralized influence of selected grouping levels.

Note

Please note that in its present form, the exclude.influence function only works on mixed effects regression models of class mer that have been estimated using the functions in the lme4 package.

Also, it is required that the mer model was estimated using a factor variable to indicate group levels. When using something similar to + (1 | as.factor(variable)), the function is not able of identifying the correct grouping factors, and returns an error.

4.1.3 ME.cook()

Compute the Cook's distance measure of influential data on mixed effects models. Cook's Distance is a measure indicating to what extent model parameters are influenced by (a set of) influential data on which the model is based. This function computes the Cook's distance based on the information returned by the estex() function.

Usage and Arguments

```
ME.cook(estex, parameters = 0, plot=FALSE, sort=FALSE, ...)
```

estex An object as returned by the estex() function, containing the altered estimates of a mixed effects regression model

parameters Used to define a selection of parameters. If parameters=0 (default), Cook's Distance is calculated based on all parameters in the model

plot If plot=TRUE, the results from the ME.cook() function are forwarded to the dp.ME.cook() function, which creates a visual representation of the Cook's Distances.

sort If sort=TRUE the values of Cook's Distance are ordered based on magnitude. If sort=FALSE (default) no sorting takes place.

... Further arguments passed on to the dp.ME.cook() function.

Value

A one-column matrix is returned containing values for the Cook's Distance based on the selected (fixed) parameters of the model. Each row shows the Cook's Distance associated with each evaluated set of influential data (data nested within each evaluated level of the grouping factor).

Examples

```
data(school23)
model <- lmer(math ~ structure + SES + (1 | school.ID), data=school23)
alt.est <- estex(model, "school.ID")
ME.cook(alt.est)
ME.cook(alt.est, plot=TRUE, cutoff=.17)
```

4.1.4 ME.dfbetas()

Compute the DFBETAS measure of influential data. DFBETAS (standardized difference of the beta) is a measure that standardizes the absolute difference in parameter estimates between a (mixed effects) regression model based on a full set of data, and a model from which a (potentially influential) subset of data is removed. A value for DFBETAS is calculated for each parameter in the model separately. This function computes the DFBETAS based on the information returned by the estex() function.

Usage and Arguments

```
ME.dfbetas(estex, parameters = 0, plot=FALSE,
    sort=FALSE, to.sort=NA, abs=FALSE, ...)
```

estex An object as returned by the estex() function, containing the altered estimates of a mixed effects regression model

parameters Used to define a selection of parameters. If parameters=0 (default), DFBETAS is calculated for all parameters in the model

plot If plot=TRUE, the results from the ME.dfbetas() function are forwarded to the dp.ME.dfbetas() function, which creates a visual representation of the values for DFBETAS

sort If sort=TRUE the values of DFBETAS are ordered based on magnitude. If sort=FALSE (default) no sorting takes place.

to.sort Specify on which variable the DFBETAS must be sorted. If only one variable present (either in the model, or due to the selection specified in parameters), this parameter can be omitted. If DFBETAS is calculated for multiple variables, and sort=TRUE, specification of to.sort is required, or an error is returned.

abs If abs=TRUE, the absolute values of DFBETAS are returned, while if abs=FALSE (default), both positive and negative values are possible. If both abs=TRUE and sort=TRUE, the abs parameters precedes the sort parameter, and thus the absolute values of DFBETAS are sorted.

... Further arguments passed on to the dp.ME.dfbetas() function.

Returned Value

A matrix is returned, containing DFBETAS-values for each (selected) fixed parameter of the model, and separately for each evaluated set of influential data.

Examples

```
data(school23)
model <- lmer(math ~ structure + SES  + (1 | school.ID), data=school23)
alt.est <- estex(model, "school.ID")
ME.dfbetas(alt.est)
ME.dfbetas(alt.est, plot=TRUE, layout=c(1,3))
```

ME.pchange

Compute the percentile change, as measure of influential data. This unstandardized measure can serve to help interpret the magnitude of the influence single or combined grouping levels exert on mixed effects models. The percentage change in parameter estimates between a (mixed effects) regression model based on a full set of data, and a model from which a (potentially influential) subset of data is removed. A value of percentage change is calculated for each

parameter in the model separately, based on the information returned by the estex() function.

Usage and arguments

```
ME.pchange(estex, parameters = 0, plot=FALSE,
    sort=FALSE, to.sort=NA, abs=FALSE, ...)
```

estex An object as returned by the estex() function, containing the altered estimates of a mixed effects regression model

parameters Used to define a selection of parameters. If parameters=0 (default), percentage change are calculated for all parameters in the model

plot If plot=TRUE, the results from the ME.pchange() function are forwarded to the dp.ME.pchange() function, which creates a visual representation of the percentage changes.

sort If sort=TRUE the values of percentage change are ordered based on magnitude. If sort=FALSE (default) no sorting takes place.

to.sort Specify on which variable the percentage changes must be sorted. If only one variable present (either in the model, or due to the selection specified in parameters), this parameter can be omitted. If percentage changes are calculated for multiple variables, and sort=TRUE, specification of to.sort is required, or an error is returned.

abs If abs=TRUE, the absolute values of percentage change are returned, while if abs=FALSE (default), both positive and negative values are possible. If both abs=TRUE and sort=TRUE, the abs parameters precedes the sort parameter, and thus the absolute values of percentage change are sorted. ... Further arguments passed on to the dp.ME.pchange() function. Value

Returned Value

A matrix is returned, containing values of percentage change for each (selected) fixed parameter estimate of the model, and separately for each evaluated set of influential data.

Examples

```
data(school23)
model <- lmer(math ~ structure + SES + (1 | school.ID), data=school23)
alt.est <- estex(model, "school.ID")
ME.pchange(alt.est)
ME.pchange(alt.est, plot=TRUE, layout=c(1,3))
```

4.2 Auxiliary Functions

4.2.1 dp.ME.cook()

Dotplot visualization of Cook's Distance. This is a wrapper function to the dotplot() function in the lattice-package. It transforms the output from the ME.cook() function and calls dotplot() to provide the user with a visualization of the Cook's Distance.

Usage and Arguments

```
dp.ME.cook(estex, parameters = 0, groups = 0, tol = 0, ...)
```

estex An object as returned by the estex() function, containing the altered estimates of a mixed effects regression model.

parameters Used to define a selection of parameters. If parameters = 0 (default), Cook's Distance is calculated based on all parameters in the model.

groups Used to define a selection of nesting groups that should be visualized. If groups = 0 (default), the values of Cook's Distance for all nesting groups are shown.

tol Values of Cook's Distance exceeding the specified level of tolerance (tol) are plotted visually different from values not exceeding tolerance. If tol=0 (default), no such differentiation is made in the way values are plotted.

... Further arguments passed on to the dotplot() function.

Examples

```
model <- lmer(diameter ~ (1|plate) + (1|sample), Penicillin)
alt.est <- estex(model, "plate")
dp.ME.cook(alt.est)
dp.ME.cook(alt.est, tol=.0015)
```

4.2.2 dp.ME.dfbetas()

Dotplot visualization of DFBETAS. This is a wrapper function to the dotplot() function in the lattice-package. It transforms the output from the ME.dfbetas() function and calls dotplot() to provide the user with a visualization of the DFBETAS.

Usage and Arguments

```
dp.ME.dfbetas(estex, parameters = 0, groups = 0, ...)
```

estex An object as returned by the estex() function, containing the altered estimates of a mixed effects regression model.

parameters Used to define a selection of parameters. If parameters = 0 (default), values for DFBETAS are visualized for parameters in the model.

groups Used to define a selection of nesting groups that should be visualized. If groups = 0 (default), the values of DFBETAS for all nesting groups are shown.

... Further arguments passed on to the dotplot() function.

Examples

```
model <- lmer(diameter ~ (1|plate) + (1|sample), Penicillin)
alt.est <- estex(model, "plate")
dp.ME.dfbetas(alt.est)
```

dp.ME.pchange

This is a wrapper function to the dotplot() function in the lattice-package. It transforms the output from the ME.pchange() function and calls dotplot() to provide the user with a visualization of the percentile changes in mixed models due to the neutralization of the influence of grouping factors.

Usage and arguments

```
dp.ME.pchange(estex, parameters = 0, groups = 0,
    sort=FALSE, to.sort=NA, abs=FALSE, ...)
```

estex An object as returned by the estex() function, containing the altered estimates of a mixed effects regression model.

parameters Used to define a selection of parameters. If parameters = 0 (default), values for percentile change are visualized for parameters in the model.

groups Used to define a selection of nesting groups that should be visualized. If groups = 0 (default), the percentile changes for all nesting groups are shown.

sort If sort=TRUE the values of percentile change are ordered based on magnitude before visualization. If sort=FALSE (default) no sorting takes place.

to.sort Specify on which variable the percentile changes must be sorted. If only one variable present (either in the model, or due to the selection specified in parameters), this parameter can be omitted. If multiple variables are visualized, and sort=TRUE, specification of to.sort is required, or an error is returned.

abs If abs=TRUE, the absolute values of percentile change are visualized, while if abs=FALSE (default), both positive and negative percentages are possible. If both abs=TRUE and sort=TRUE, the abs parameters precedes the sort parameter, and thus the absolute values of percentile change are sorted. ... Further arguments passed on to the dotplot() function. Author(s)

Examples

```
data(school23)
model <- lmer(math ~ structure + SES  + (1 | school.ID), data=school23)
alt.est <- estex(model, "school.ID")
dp.ME.pchange(alt.est)
dp.ME.pchange(alt.est, layout=c(1,3))
```

4.2.3 grouping.levels()

Returns the levels of a grouping factor in a mixed effects regression model. Helper function returning all the levels of a grouping factor in a mixed effects regression model. This is used in combination with the influence.ME function, the divide long runs into multiple shorter ones.

Usage and Arguments

```
grouping.levels(model, group)
```

model Mixed effects model of class 'mer'

group Grouping factor of 'model' of which the levels are returned

Details

Please note that at times different results may be obtained by using nesting.levels(), compared with deriving the levels of the grouping factor directly from the (original) data. This is because nesting.levels() only extracts the nesting levels that were de facto used in the model. Due to missing values, this may diverge from those present in the actual data.

Returned Value

Returns a character vector containing all the names / labels of levels of the grouping factor.

Note

Admittedly, this function is of little use in the 0.5 version of the influence.ME package. However, it will be used in future versions, in which functionality will be provided to divide long influence.ME runs over different sessions or even computers.

Examples

```
model <- lmer(diameter ~ (1|plate) + (1|sample), Penicillin)
grouping.levels(model, "plate")
grouping.levels(model, "sample")
```

4.2.4 The school23 data

The `school23` data contains information on students' performance on a math test, as well as several explanatory variables. These data are subset of the NELS-88 data (National Education Longitudinal Study of 1988). Both a selected number of variables and a selected number of observations are given here.

Usage and Variables

```
data(school23)
```

A data frame with 519 observations on the following 15 variables.

school.ID a factor with 23 levels, representing the 23 schools within which students are nested.

SES a numeric vector, representing the socio-economic status

mean.SES a numeric vector, representing the mean socio-economic status per school

homework a factor representing the time spent on math homework each week, with levels None, Less than 1 hour, 1 hour, 2 hours, 3 hours, 4-6 hours, 7-9 hours, and 10 or more

parented a factor representing the parents' highest education level, with levels Dod not finish H.S., H.S. grad or GED, GT H.S. and LT 4yr degree, College graduate, M.A. or equivalent, and Ph.D., M.D., other

ratio a numeric vector, representing the student-teacher ratio perc.minor a factor representing the percent minority in school, with levels None, 1-5, 6-10, 11-20, 21-40, 41-60, 61-90, and 91-100

math a numeric vector, representing the number of correct answers on a mathematics test sex a factor with levels Male and Female race a factor with levels Asian, Hispanic, Black, White, and American Indian

school.type a factor representing the school type, with levels Public school, Catholic school, Private, other religious affiliation, and Private, no religious affiliation

structure a numeric vector representing the degree to which the classroom environment is structured. High values represent higher levels of (accurate) classroom environment structure

school.size a factor representing the total school enrollment, with levels 1-199 Students, 200-399, 400-599, 600-799, 800-999, 1000-1199, and 1200+

urban a factor with levels Urban, Suburban, and Rural region a factor with levels Northeast, North Central, South, and West Details

Source

Labels for the factors were found in an appendix in Kreft & De Leeuw(1998). All labels were designated, although in some cases not all possible values are represented in the variable (i.e. region). This is probably due to the fact that this is only a subsample from the full NELS-88 data(Kreft and De Leeuw, 1998).

Also, some of the variable names were changed.

These data are used in the examples given in Kreft & De Leeuw (1998). Both the examples and the data are publicly available from the internet:

http://www.ats.ucla.edu/stat/examples/imm/. Data were reproduced with kind permission from Jan de Leeuw.

Examples

```
data(school23)
model <- lmer(math ~ structure + (1 | school.ID), data=school23)
```

Bibliography

Bates, D., Maechler, M., and Dai, B. (2008). *lme4: Linear mixed-effects models using S4 classes*. R package version 0.999375-28.

Belsley, D. A., Kuh, E., and Welsch, R. E. (1980). *Regression Diagnostics. Identifying Influential Data and Sources of Collinearity*. Wiley.

Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18.

Fox, J. (2002). *An R and S-Plus Companion to Applied Regression*. Sage.

Gelman, A. and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical methods for social research. Cambridge University Press.

Kreft, I. and De Leeuw, J. (1998). *Introducing Multilevel Modeling*. Sage Publications.

Langford, I. H. and Lewis, T. (1998). Outliers in multilevel data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 161:121–160.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Sarkar, D. (2008). *Lattice. Multivariate Data Visualization with R*. Springer.

Snijders, T. A. and Berkhof, J. (2008). Diagnostic checks for multilevel models. In De Leeuw, J. and Meijer, E., editors, *Handbook of Multilevel Analysis*, chapter 3, pages 141–175. Springer.

Snijders, T. A. and Bosker, R. J. (1999). *Multilevel Analysis, an introduction to basic and advanced multilevel modelling*. Sage.

Van der Meer, T., te Grotenhuis, M., and Pelzer, B. (2009). Influential cases in multilevel modeling. a methodological comment on ruiter and de graaf (asr, 2006). *American Sociological Review*, Accepted for Publication.