Creating an R data set from STAR

Douglas Bates
Department of Statistics
University of Wisconsin – Madison

April 5, 2005

Abstract

A substantial portion of the data from Tennessee's Student Teacher Achievement Ratio (STAR) project, a large-scale, four-year study of reduced class size, has been made available to the public at `http://www.heros-inc.org/data.htm`. We describe the creation of an R (`http:www.r-project.org`) data set from these data.

1 Introduction

The data from the STAR project are available in several different forms from the web site `http://www.heros-inc.org/data.htm`. The most convenient form for creation of an R data set is the tab-delimited text file. Download the archive file `http://www.heros-inc.org/text-star.zip` containing two files: `readme.txt`, a description of the data, and `webstar.txt`, the data themselves.

2 Reading the data

From the data description file we can see that there are 53 columns in the data set and most of these columns are coded values. Such columns should be represented as factors in R but many of these columns will need to be combined before we can work with them. We will convert the first 5 columns to factors and leave the remaining 48 columns as integers.

```
> str(orig <- read.delim(unz(system.file("original/text-star.zip", package = "mlmRev"),
+                              "webstar.txt"), colCl = rep(c("factor", "integer"), c(5, 48))))

'data.frame':        11598 obs. of  53 variables:
 $ NEWID   : Factor w/ 11598 levels "100017","100028",..: 839 943 986 1104 1263 1346 1485 15
 $ SSEX    : Factor w/ 3 levels "1","2","9": 2 2 2 1 1 1 1 2 1 1 ...
 $ SRACE   : Factor w/ 7 levels "1","2","3","4",..: 2 1 2 1 2 1 2 1 1 1 ...
 $ SBIRTHQ : Factor w/ 5 levels "1","2","3","4",..: 3 1 4 4 1 3 1 4 2 3 ...
 $ SBIRTHY : Factor w/ 7 levels "1977","1978",..: 3 4 3 3 4 3 3 3 3 3 ...
```

```
$ STARK   : int  2 1 1 2 1 2 2 2 2 2 ...
$ STAR1   : int  2 1 1 2 2 2 2 1 1 1 ...
$ STAR2   : int  2 1 1 2 2 1 2 1 1 1 ...
$ STAR3   : int  1 1 1 1 2 1 1 1 1 1 ...
$ CLTYPEK : int  9 1 1 9 3 9 9 9 9 9 ...
$ CLTYPE1 : int  9 1 1 9 9 9 9 3 2 2 ...
$ CLTYPE2 : int  9 1 3 9 9 2 9 3 2 2 ...
$ CLTYPE3 : int  2 1 3 1 9 2 3 3 2 2 ...
$ SCHTYPEK: int  9 3 2 9 1 9 9 9 9 9 ...
$ HDEGK   : int  9 2 2 9 2 9 9 9 9 9 ...
$ CLADK   : int  9 1 1 9 6 9 9 9 9 9 ...
$ TOTEXPK : int  99 7 21 99 0 99 99 99 99 99 ...
$ TRACEK  : int  9 1 1 9 1 9 9 9 9 9 ...
$ TREADSSK: int  999 447 450 999 439 999 999 999 999 999 ...
$ TMATHSSK: int  999 473 536 999 463 999 999 999 999 999 ...
$ SESK    : int  9 2 2 9 1 9 9 9 9 9 ...
$ SCHTYPE1: int  9 3 2 9 9 9 9 3 3 3 ...
$ TRACE1  : int  9 1 2 9 9 9 9 1 1 1 ...
$ HDEG1   : int  9 1 2 9 9 9 9 2 2 1 ...
$ CLAD1   : int  9 4 3 9 9 9 9 2 4 4 ...
$ TOTEXP1 : int  99 7 32 99 99 99 99 8 13 7 ...
$ TREADSS1: int  999 507 579 999 999 999 999 475 999 651 ...
$ TMATHSS1: int  999 538 592 999 999 999 999 512 999 532 ...
$ SES1    : int  9 1 9 9 9 9 9 2 2 2 ...
$ SCHTYPE2: int  9 3 2 9 9 3 9 3 3 3 ...
$ TRACE2  : int  9 1 2 9 9 1 9 2 1 1 ...
$ HDEG2   : int  9 1 1 9 9 1 9 2 1 1 ...
$ CLAD2   : int  9 2 4 9 9 1 9 4 4 4 ...
$ TOTEXP2 : int  99 3 4 99 99 13 99 13 6 8 ...
$ TREADSS2: int  999 568 588 999 999 999 999 573 999 596 ...
$ TMATHSS2: int  999 579 579 999 999 999 999 550 999 590 ...
$ SES2    : int  9 2 2 9 9 2 9 2 2 2 ...
$ SCHTYPE3: int  2 3 2 3 9 3 1 3 3 3 ...
$ TREADSS3: int  580 587 644 686 999 644 999 599 999 626 ...
$ TMATHSS3: int  564 593 639 667 999 648 999 583 999 618 ...
$ SES3    : int  1 1 2 2 9 2 1 2 2 2 ...
$ TRACE3  : int  1 1 1 1 9 1 2 1 1 1 ...
$ HDEG3   : int  1 1 1 1 9 1 1 1 1 2 ...
$ CLAD3   : int  4 2 4 4 9 4 1 6 4 4 ...
$ TOTEXP3 : int  30 1 4 10 99 15 17 23 8 8 ...
$ SYSIDKN : int  999 30 11 999 11 999 999 999 999 999 ...
$ SYSID1N : int  999 30 11 999 999 999 999 4 40 21 ...
$ SYSID2N : int  999 30 11 999 999 6 999 4 40 21 ...
$ SYSID3N : int  22 30 11 6 999 6 11 4 40 21 ...
$ SCHIDKN : int  999 63 20 999 19 999 999 999 999 999 ...
$ SCHID1N : int  999 63 20 999 999 999 999 5 77 50 ...
```

```
 $ SCHID2N : int  999 63 20 999 999 8 999 5 77 50 ...
 $ SCHID3N : int  54 63 20 8 999 8 31 5 77 50 ...
```

In the call to `read.delim` we read directly from the zip archive and avoided expanding the much larger text file. The call to `system.file` determines the name of a file that is part of a package. In practice it is often more convenient to use the `file.choose` function which brings up a file chooser panel. load(system.file("original/orig.rda", package = "mlmRev"))

We can check the form of the original data with

```
> str(orig)
```

```
'data.frame':        11598 obs. of  53 variables:
 $ NEWID   : Factor w/ 11598 levels "100017","100028",..: 839 943 986 1104 1263 1346 1485 15
 $ SSEX    : Factor w/ 3 levels "1","2","9": 2 2 2 1 1 1 1 2 1 1 ...
 $ SRACE   : Factor w/ 7 levels "1","2","3","4",..: 2 1 2 1 2 1 2 1 1 1 ...
 $ SBIRTHQ : Factor w/ 5 levels "1","2","3","4",..: 3 1 4 4 1 3 1 4 2 3 ...
 $ SBIRTHY : Factor w/ 7 levels "1977","1978",..: 3 4 3 3 4 3 3 3 3 3 ...
 $ STARK   : int  2 1 1 2 1 2 2 2 2 2 ...
 $ STAR1   : int  2 1 1 2 2 2 2 1 1 1 ...
 $ STAR2   : int  2 1 1 2 2 1 2 1 1 1 ...
 $ STAR3   : int  1 1 1 1 2 1 1 1 1 1 ...
 $ CLTYPEK : int  9 1 1 9 3 9 9 9 9 9 ...
 $ CLTYPE1 : int  9 1 1 9 9 9 9 3 2 2 ...
 $ CLTYPE2 : int  9 1 3 9 9 2 9 3 2 2 ...
 $ CLTYPE3 : int  2 1 3 1 9 2 3 3 2 2 ...
 $ SCHTYPEK: int  9 3 2 9 1 9 9 9 9 9 ...
 $ HDEGK   : int  9 2 2 9 2 9 9 9 9 9 ...
 $ CLADK   : int  9 1 1 9 6 9 9 9 9 9 ...
 $ TOTEXPK : int  99 7 21 99 0 99 99 99 99 99 ...
 $ TRACEK  : int  9 1 1 9 1 9 9 9 9 9 ...
 $ TREADSSK: int  999 447 450 999 439 999 999 999 999 999 ...
 $ TMATHSSK: int  999 473 536 999 463 999 999 999 999 999 ...
 $ SESK    : int  9 2 2 9 1 9 9 9 9 9 ...
 $ SCHTYPE1: int  9 3 2 9 9 9 9 3 3 3 ...
 $ TRACE1  : int  9 1 2 9 9 9 9 1 1 1 ...
 $ HDEG1   : int  9 1 2 9 9 9 9 2 2 1 ...
 $ CLAD1   : int  9 4 3 9 9 9 9 2 4 4 ...
 $ TOTEXP1 : int  99 7 32 99 99 99 99 8 13 7 ...
 $ TREADSS1: int  999 507 579 999 999 999 999 475 999 651 ...
 $ TMATHSS1: int  999 538 592 999 999 999 999 512 999 532 ...
 $ SES1    : int  9 1 9 9 9 9 9 2 2 2 ...
 $ SCHTYPE2: int  9 3 2 9 9 3 9 3 3 3 ...
 $ TRACE2  : int  9 1 2 9 9 1 9 2 1 1 ...
 $ HDEG2   : int  9 1 1 9 9 1 9 2 1 1 ...
 $ CLAD2   : int  9 2 4 9 9 1 9 4 4 4 ...
 $ TOTEXP2 : int  99 3 4 99 99 13 99 13 6 8 ...
```

```
 $ TREADSS2: int  999 568 588 999 999 999 999 573 999 596 ...
 $ TMATHSS2: int  999 579 579 999 999 999 999 550 999 590 ...
 $ SES2    : int  9 2 2 9 9 2 9 2 2 2 ...
 $ SCHTYPE3: int  2 3 2 3 9 3 1 3 3 3 ...
 $ TREADSS3: int  580 587 644 686 999 644 999 599 999 626 ...
 $ TMATHSS3: int  564 593 639 667 999 648 999 583 999 618 ...
 $ SES3    : int  1 1 2 2 9 2 1 2 2 2 ...
 $ TRACE3  : int  1 1 1 1 9 1 2 1 1 1 ...
 $ HDEG3   : int  1 1 1 1 9 1 1 1 1 2 ...
 $ CLAD3   : int  4 2 4 4 9 4 1 6 4 4 ...
 $ TOTEXP3 : int  30 1 4 10 99 15 17 23 8 8 ...
 $ SYSIDKN : int  999 30 11 999 11 999 999 999 999 999 ...
 $ SYSID1N : int  999 30 11 999 999 999 999 4 40 21 ...
 $ SYSID2N : int  999 30 11 999 999 6 999 4 40 21 ...
 $ SYSID3N : int  22 30 11 6 999 6 11 4 40 21 ...
 $ SCHIDKN : int  999 63 20 999 19 999 999 999 999 999 ...
 $ SCHID1N : int  999 63 20 999 999 999 999 5 77 50 ...
 $ SCHID2N : int  999 63 20 999 999 8 999 5 77 50 ...
 $ SCHID3N : int  54 63 20 8 999 8 31 5 77 50 ...
```

2.1 Missing value codes

All the columns except the first column have missing values present. Typically the missing value code is `"9"` but `"99"`, `"999"` and `"9999"` are also used. We convert these to R's missing value code `NA` column by column.

```
> mv <- rep("9", 53)
> mv[c(4,17,26,34,45)] <- "99"
> mv[c(19,20,27,28,35,36,39,40,46:53)] <- "999"
> mv[5] <- "9999"
> mv[1] <- "999999"
> for (i in seq(a = orig)) orig[[i]][orig[[i]] == mv[i]] <- NA
> summary(orig[1:5])
```

```
     NEWID          SSEX           SRACE       SBIRTHQ   
 100017 :    1   1   :6122   1       :7193   1   :2836  
 100028 :    1   2   :5456   2       :4173   2   :2851  
 100045 :    1   9   :   0   3       :  32   3   :3422  
 100064 :    1   NA's:  20   4       :  21   4   :2423  
 100070 :    1               6       :  20   99  :   0  
 100096 :    1               (Other):  14   NA's:  66  
 (Other):11592               NA's   : 145              
   SBIRTHY  
 1980   :6886  
 1979   :3915  
 1978   : 645  
 1977   :  58  
```

```
 1981   :  24
 (Other):   1
 NA's   :  69
```

Notice that level "9" is still present for the SSEX variable even after all the observations at that level have been replaced by the missing value code. To remove these unused levels from this and all the other columns, we loop over the columns selecting all the values but using the optional argument `drop = TRUE`.

```
> for (i in seq(a = orig)) orig[[i]] <- orig[[i]][drop = TRUE]
> summary(orig[1:5])
```

```
     NEWID           SSEX       SRACE       SBIRTHQ     SBIRTHY
 100017 :    1   1   :6122   1   :7193   1   :2836   1977:  58
 100028 :    1   2   :5456   2   :4173   2   :2851   1978: 645
 100045 :    1   NA's:  20   3   :  32   3   :3422   1979:3915
 100064 :    1               4   :  21   4   :2423   1980:6886
 100070 :    1               5   :  14   NA's:  66   1981:  24
 100096 :    1               6   :  20               1982:   1
 (Other):11592               NA's: 145               NA's:  69
```

For convenience we convert the names of the columns to lower case.

```
> names(orig) <- tolower(names(orig))
```

3 Setting factor levels

In R the levels of a factor can be given meaningful labels instead of numeric codes and in most cases this eliminates the need for a separate codebook. For example storing the labels of `sex` as "M" and "F" makes the coding self-explanatory. When used in a model a factor is automatically converted to a set of "contrasts" (there is a technical definition of the term "contrast" in linear models that is not always fulfilled by these derived variables) and the corresponding coefficients are given meaningful names.

When there is a natural ordering of the levels of a factor it can be created as an ordered factor that will preserve this ordering.

The labels can be set after the factor is created or as part of the creation of the factor. Below we will create a "long form" of the data where each row corresponds to a combination of student and grade. In doing this we will need to concatenate related columns of the original data frame. For example, the columns `cltypek`, `cltype1`, `cltype2` and `cltype3` will be concatenated to form a single column `cltype`. If the coding is consistent across the grades then it is easiest to concatenate the integer codes and set the labels on the "long" version of the variable.

However there are two groups of variables, `hdeg` and `clad`, that are not coded consistently. In each case the codes used for kindergarten teachers are different from those used for teachers of grades 1 to 3 classes. The codes for kindergarten teachers are a superset of those for the other teachers but the numbering is not consistent; a bachelor's degree is coded as 2 for kindergarten but 1 for the others. Thus we cannot combine the numeric values - we must create the labels for each column and then concatenate the labels and convert to a factor.

```
> orig$hdegk <- ordered(orig$hdegk, levels = 1:6,
+                         labels = c("ASSOC","BS/BA","MS/MA/MEd","MA+","Ed.S","Ed.D/Ph.D"))
> orig$hdeg1 <- ordered(orig$hdeg1, levels = 1:4,
+                         labels = c("BS/BA","MS/MA/MEd","Ed.S","Ed.D/Ph.D"))
> orig$hdeg2 <- ordered(orig$hdeg2, levels = 1:4,
+                        labels = c("BS/BA","MS/MA/MEd","Ed.S","Ed.D/Ph.D"))
> orig$hdeg3 <- ordered(orig$hdeg3, levels = 1:4,
+                        labels = c("BS/BA","MS/MA/MEd","Ed.S","Ed.D/Ph.D"))
> orig$cladk <- factor(orig$cladk, levels = c(1:3,5:8),
+                        labels = c("1","2","3","APPR","PROB","NOT","PEND"))
> orig$clad1 <- factor(orig$clad1, levels = 1:6,
+                        labels = c("NOT","APPR","PROB","1","2","3"))
> orig$clad2 <- factor(orig$clad2, levels = 1:6,
+                        labels = c("NOT","APPR","PROB","1","2","3"))
> orig$clad3 <- factor(orig$clad3, levels = 1:6,
+                        labels = c("NOT","APPR","PROB","1","2","3"))
```

4 Creating separate data frames

These data are represented in a "wide" format where each row corresponds to a student. Some of the columns, such as `ssex`, are indeed a property of the student; some, such as `hdegk` are properties of teachers; some, such as `schtypek` are properties of schools or classes in schools; and some are unique to a student/grade combination. We will create separate frames for each of these types.

The first 5 columns are student-level data

```
> student <- orig[1:5]
> names(student) <- c("id", "sx", "eth", "birthq", "birthy")
> levels(student$sx) <- c("M", "F")
> levels(student$eth) <- c("W", "B", "A", "H", "I", "O")
> student$birthy <- ordered(student$birthy)
> student$birthq <- ordered(paste(student$birthy,student$birthq,sep=":"))
> summary(student)
```

```
       id           sx         eth          birthq       birthy
 100017 :    1   M   :6122   W   :7193   1980:3 :2304   1977:  58
 100028 :    1   F   :5456   B   :4173   1980:1 :2221   1978: 645
 100045 :    1   NA's:  20   A   :  32   1980:2 :2190   1979:3915
 100064 :    1               H   :  21   1979:4 :1879   1980:6886
 100070 :    1               I   :  14   1979:3 : 923   1981:  24
```

```
 100096 :    1               0    :  20   1979:2 : 586   1982:   1
 (Other):11592               NA's: 145   (Other):1495   NA's:  69
```

The other columns refer to a combination of the student and grade. We first create an expanded or "long" version of the table with a row for each student/grade combination.

To create the long version of the table we repeat the student ids four times and add a column for the grade level. Related groups of columns, such as `cltypek`, `cltype1`, `cltype2` and `cltype3`, are concatenated then converted to a factor. However, there are two groups, `hdeg` and `clad`, for which this approach will not work because these groups are not encoded consistently.

```
> long <- data.frame(id = rep(orig$newid, 4),
+                    gr = ordered(rep(c("K", 1:3), each = nrow(orig)),
+                                 levels = c("K", 1:3)),
+                    star = factor(unlist(orig[6:9])),
+                    cltype = factor(unlist(orig[10:13])),
+                    schtype = factor(unlist(orig[c(14,22,30,38)])),
+                    hdeg = ordered(unlist(lapply(orig[c(15,24,32,43)],as.character)),
+                                   levels = c("ASSOC","BS/BA","MS/MA/MEd","MA+","Ed.S","Ed.
+                    clad = factor(unlist(lapply(orig[c(16,25,33,44)],as.character)),
+                                   levels = c("NOT","APPR","PROB","PEND","1","2","3")),
+                    exp = unlist(orig[c(17,26,34,45)]),
+                    trace = factor(unlist(orig[c(18,23,31,42)]), levels=1:6,
+                                   labels=c("W", "B", "A", "H", "I", "O")),
+                    read = unlist(orig[c(19,27,35,39)]),
+                    math = unlist(orig[c(20,28,36,40)]),
+                    ses = factor(unlist(orig[c(21,29,37,41)]),labels=c("F","N")),
+                    sch = factor(unlist(orig[50:53])))
```

We can now eliminate the combinations that are completely missing. Checking

```
> summary(long)
```

```
       id          gr          star        cltype       schtype
 100017 :    4   K:11598   1:26796   1    : 8015   1    : 5624
 100028 :    4   1:11598   2:19596   2    : 9192   2    : 6428
 100045 :    4   2:11598             3    : 9589   3    :12561
 100064 :    4   3:11598             NA's:19596   4    : 2183
 100070 :    4                                    NA's:19596
 100096 :    4
 (Other):46368
        hdeg            clad           exp             trace
 ASSOC    :    0   1      :18303   Min.   :    0.00   W    :21550
 BS/BA    :16586   APPR   : 2030   1st Qu.:    5.00   B    : 5005
 MS/MA/MEd: 9587   PROB   : 1961   Median :   11.00   A    :   14
```

```
 MA+      :  161   NOT    : 1757   Mean   :   12.04   H   :    0
 Ed.S     :  237   3      : 1059   3rd Qu.:   17.00   I   :    0
 Ed.D/Ph.D:   58   (Other):  877   Max.   :   42.00   O   :    0
 NA's     :19763   NA's   :20405   NA's   :19789.00   NA's:19823
      read            math            ses            sch
 Min.   : 315   Min.   : 288.0   F   :13111   51     :  826
 1st Qu.: 467   1st Qu.: 505.0   N   :12858   27     :  562
 Median : 552   Median : 557.0   NA's:20423   9      :  543
 Mean   : 540   Mean   : 553.7                22     :  534
 3rd Qu.: 604   3rd Qu.: 603.0                63     :  534
 Max.   : 775   Max.   : 774.0                (Other):23797
 NA's   :22130  NA's   :21779.0               NA's   :19596
```

indicates that fewest missing values are in the `sch`, `cltype`, and `schtype` columns. They are also consistent

```
> with(long, all.equal(is.na(schtype), is.na(sch)))
```

```
[1] TRUE
```

```
> with(long, all.equal(is.na(cltype), is.na(sch)))
```

```
[1] TRUE
```

hence we use these to subset the data frame

```
> long <- long[!is.na(long$sch),]
```

It turns out that we could have used the `star` column as this simply indicates if the student was in the study that year.

```
> summary(long[1:5])
```

```
       id          gr         star         cltype    schtype
 100173 :    4   K:6325   1:26796   1:8015   1: 5624
 100201 :    4   1:6829   2:    0   2:9192   2: 6428
 10023  :    4   2:6840             3:9589   3:12561
 100236 :    4   3:6802                      4: 2183
 100302 :    4
 100361 :    4
 (Other):26772
```

Because it now contains no information we will drop it.

```
> long$star <- NULL
```

For convenience we set the row names of this data frame to be a combination of the student id and the grade.

```
> rownames(long) <- paste(long$id, long$gr, sep = '/')
```

We can extract the school-level data from this table.

```
> school <- unique(long[, c("sch", "schtype")])
> length(levels(school$sch)) == nrow(school)

[1] TRUE

> row.names(school) <- school$sch
> school <- school[order(as.integer(as.character(school$sch))),]
> long$schtype <- NULL
> levels(school$schtype) <- c("inner","suburb","rural","urban")
> levels(long$cltype) <- c("small", "reg", "reg+A")
```

We can create a merged data set with

```
> star <- merge(merge(long, school, by = "sch"), student, by = "id")
> star$time <- as.integer(star$gr) - 1
```

5 Assigning teacher ids

There are no teacher id numbers available but we can obtain a reasonably accurate surrogate by determining the unique combinations of all the variables associated with the teacher.

```
> teacher <- unique(star[, c("cltype", "trace", "exp", "clad", "gr",
+                            "hdeg", "sch")])
> teacher <- teacher[with(teacher, order(sch, gr, cltype, exp, hdeg,
+                                        clad, trace)), ]
```

To generate the correspondence between the observations and the teacher we create labels that incorporate the levels of each of the variables that defined the unique combinations.

```
> row.names(teacher) <- tch <- teacher$tch <- seq(nrow(teacher))
> names(tch) <- do.call("paste", c(teacher[,1:7], list(sep=":")))
> star$tch <- tch[do.call("paste", c(star[c("cltype", "trace", "exp",
+                                            "clad", "gr", "hdeg", "sch")],
+                                    list(sep = ":")))]
```

We can check if this is successful by generating tables of class sizes.

```
> table(table(star$tch))

  1   2   3  11  12  13  14  15  16  17  18  19  20  21  22  23  24
 45   4   1   2  17  68  81 116 111 106  28  41  49 103 152 137 138
 25  26  27  28  29  30  32  44  46
 83  47  29  13  11   2   1   1   1

> table(table(subset(star, cltype == "small")$tch))
```

```
  1   2  11  12  13  14  15  16  17  18  19  20  32
 12   1   2  17  68  81 114 108 100  22   9   1   1

> table(table(subset(star, cltype == "reg")$tch))

 1  2 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
21  1  1  2  5  4 16 31 47 84 64 70 38 20 11  4  5  1

> table(table(subset(star, cltype == "reg+A")$tch))

 1  2  3 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 44 46
12  2  1  1  1  1  2 16 17 56 68 73 68 45 27 18  9  6  1  1  1
```

We see that there are three classes with sizes greater than 30 and that one of these is labelled as a "small" class. It is likely that each of these represents two or more classes but we do not have enough information to distinguish them.

6 Initial model fits

Some initial model fits are

```
> library(lme4)
> (mm1 <- lmer(math ~ gr + sx + eth + cltype + (1|id) + (1|sch), star))

Linear mixed model fit by REML
Formula: math ~ gr + sx + eth + cltype + (1 | id) + (1 | sch)
   Data: star
    AIC    BIC  logLik deviance REMLdev
 245170 245292 -122570   245173  245140
Random effects:
 Groups   Name        Variance Std.Dev.
 id       (Intercept) 982.91   31.351
 sch      (Intercept) 122.61   11.073
 Residual             686.62   26.203
Number of obs: 24578, groups: id, 10732; sch, 80

Fixed effects:
            Estimate Std. Error t value
(Intercept) 560.4976     1.4748   380.0
gr.L         96.4124     0.3961   243.4
gr.Q         -4.5797     0.3638   -12.6
gr.C         -3.4602     0.3494    -9.9
sxF           2.9526     0.7150     4.1
ethB        -22.8913     1.2748   -18.0
ethA          2.1285     7.0290     0.3
ethH          1.1732    10.1719     0.1
ethI        -34.7840    14.5073    -2.4
```

```
ethO          2.4194     8.7213     0.3
cltypereg    -7.1113     0.7280    -9.8
cltypereg+A  -5.9104     0.7398    -8.0

Correlation of Fixed Effects:
            (Intr) gr.L   gr.Q   gr.C   sxF    ethB   ethA   ethH
gr.L        -0.010
gr.Q        -0.013 -0.059
gr.C        -0.003  0.012  0.001
sxF         -0.232  0.000 -0.002  0.001
ethB        -0.268 -0.019  0.010  0.000 -0.001
ethA        -0.027 -0.001  0.000 -0.005 -0.002  0.053
ethH        -0.021  0.000  0.007 -0.001  0.004  0.024  0.003
ethI        -0.016  0.003  0.007 -0.002  0.005  0.025  0.005  0.005
ethO        -0.016  0.000  0.005 -0.003  0.008  0.019  0.007  0.005
cltypereg   -0.294  0.086  0.027 -0.015  0.005 -0.008  0.002  0.013
cltypereg+A -0.297  0.050 -0.006  0.008  0.005 -0.007  0.009  0.008
            ethI   ethO   cltypr
gr.L
gr.Q
gr.C
sxF
ethB
ethA
ethH
ethI
ethO         0.003
cltypereg    0.001 -0.003
cltypereg+A  0.007 -0.003  0.676
```

> (rm1 <- lmer(read ~ gr + sx + eth + cltype + (1|id) + (1|sch), star))

```
Linear mixed model fit by REML
Formula: read ~ gr + sx + eth + cltype + (1 | id) + (1 | sch)
   Data: star
    AIC    BIC  logLik deviance REMLdev
 241495 241616 -120732   241498  241465
Random effects:
 Groups   Name        Variance Std.Dev.
 id       (Intercept) 943.99   30.724
 sch      (Intercept) 109.48   10.463
 Residual             692.06   26.307
Number of obs: 24226, groups: id, 10621; sch, 80

Fixed effects:
            Estimate Std. Error t value
```

```
(Intercept) 541.7017     1.4155   382.7
gr.L        131.4031     0.4007   327.9
gr.Q        -28.2074     0.3678   -76.7
gr.C         -1.6300     0.3531    -4.6
sxF           9.0608     0.7095    12.8
ethB        -18.8087     1.2545   -15.0
ethA          8.5029     6.9356     1.2
ethH          1.9415    10.0431     0.2
ethI        -32.0995    14.3606    -2.2
ethO          7.6353     8.6120     0.9
cltypereg    -7.8280     0.7286   -10.7
cltypereg+A  -4.7449     0.7402    -6.4

Correlation of Fixed Effects:
            (Intr) gr.L   gr.Q   gr.C   sxF    ethB   ethA   ethH
gr.L        -0.008
gr.Q        -0.015 -0.061
gr.C        -0.001  0.014 -0.014
sxF         -0.240  0.000 -0.002  0.001
ethB        -0.275 -0.021  0.011  0.000 -0.001
ethA        -0.028 -0.001  0.000 -0.005 -0.002  0.052
ethH        -0.022  0.000  0.007 -0.002  0.004  0.024  0.003
ethI        -0.016  0.003  0.007 -0.002  0.005  0.025  0.005  0.005
ethO        -0.018  0.001  0.005 -0.003  0.008  0.019  0.007  0.005
cltypereg   -0.306  0.085  0.027 -0.014  0.005 -0.008  0.002  0.013
cltypereg+A -0.308  0.047 -0.006  0.009  0.005 -0.006  0.009  0.008
            ethI   ethO   cltypr
gr.L
gr.Q
gr.C
sxF
ethB
ethA
ethH
ethI
ethO         0.003
cltypereg    0.001 -0.002
cltypereg+A  0.007 -0.003  0.671
```

7 Session Info

```
> toLatex(sessionInfo())
```

- R version 2.9.1 (2009-06-26), `i486-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8;LC_NUMERIC=C;LC_TIME=en_US.UTF-8;LC_COLLATE=en_US.UTF-`

`8;LC_MONETARY=C;LC_MESSAGES=en_US.UTF-8;LC_PAPER=en_US.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_`
`8;LC_IDENTIFICATION=C`

- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: lattice 0.17-25, lme4 0.999375-32, Matrix 0.999375-30, mlmRev 1.0-0
- Loaded via a namespace (and not attached): grid 2.9.1