Statistical methods for linguistic research: Foundational Ideas

Audrey Bürki and Shravan Vasishth

Universität Potsdam
buerki@uni-potsdam.de

August 23, 2017

Linear models

Returning to our noise and deg data, one important point we've neglected is that different subjects may have different effects of noise and deg. In the linear models we fit we were ignoring this. This is not without consequence. It may be that our effect is driven by one or two participants (and would not generalize to other participants). Let's look at the means for the different participants.

```
noisedeg<-read.table("data/noisedeg.txt",
                     header=TRUE)
means.noise<-with(noisedeg,tapply(rt,list(subj,noise),
                                  mean))
means.deg<-with(noisedeg,tapply(rt,list(subj,
                                        deg),mean))
```

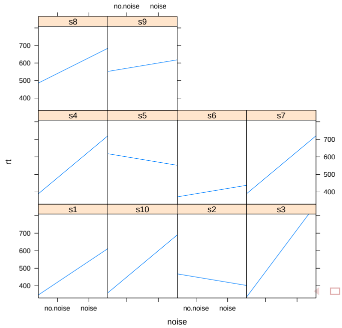
Linear models

```
head(means.noise, n=3)

##     no.noise noise
## s1       420   540
## s10      450   600
## s2       450   420

head(means.deg, n=3)

##       0   4
## s1  450 510
## s10 510 540
## s2  390 480
```

Linear models

We can visualize the different responses of the subjects for the effect of noise:

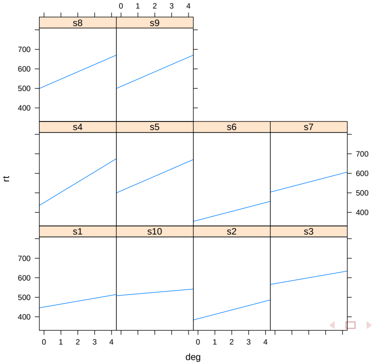
```
# library(lattice)
print(xyplot(rt~noise|subj, panel=function(x,y,...)
          {panel.xyplot(x,y,type="r")},noisedeg))
```



Linear models

We can do the same for degree:

```
print(xyplot(rt~deg|subj,
        panel=function(x,y,...)
          {panel.xyplot(x,y,type="r")},noisedeg))
```



Linear models

Given these differences between subjects, we could fit a separate linear model for each subject, collect together the intercepts and slopes for each subject, and then check if the intercepts and slopes (averages) are significantly different from zero.

Linear models

Fit a separate model for one subject (s1):

```
## fit a separate linear model for subject s1:
s1data<-subset(noisedeg,subj=="s1")
m<-lm(rt~noise,s1data)
round(summary(m)$coefficients,digits=2)

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)      420      42.43     9.9     0.01
## noisenoise       120      60.00     2.0     0.18
```

Look at the means for s1 for noise and compare them to the coefficients above.

Linear models

Now we can do this for every one of our 10 subjects.
There is a function in the package `lme4` that does this for you: `lmList`.

```
## do the same as the above for-loop for each subject:
library(lme4)

lmlist.fm1<-lmList(rt~noise|subj,noisedeg)
```

Linear models

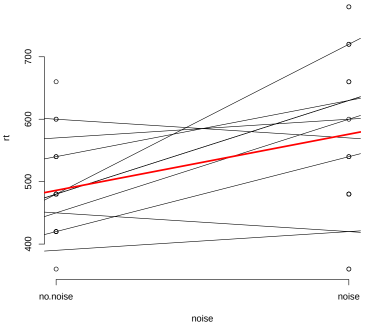
Now we can do this for every one of our 10 subjects.
There is a function in the package `lme4` that does this for you: `lmList`.

```
## do the same as the above for-loop for each subject:
library(lme4)

lmlist.fm1<-lmList(rt~noise|subj,noisedeg)
```

```
lmlist.fm1$s1$coefficients

## (Intercept)  noisenoise
##         420         120
```

Linear models

One can plot the individual lines for each subject, as well as the linear model m0's line (this shows how each subject deviates in intercept and slope from the model m0's intercept and slopes).



Linear models

To find out if there is an effect of noise, we could check whether the slopes of the individual subjects' fitted lines taken together are significantly different from zero using the t-test

Linear models

```
t.test(coef(lmlist.fm1)[2])

##
##  One Sample t-test
##
## data:  coef(lmlist.fm1)[2]
## t = 3.2225, df = 9, p-value = 0.01045
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   26.82139 153.17861
## sample estimates:
## mean of x
##        90
```

Linear models

Problems with fitting a model for each subject:

- We ignore the fact that the data points for a given subject in a given condition are coming from the same subject (they are not independent).
- We do not model the fact that the datapoints for the different subjects come from the same task/stimuli

The solution: linear mixed-effects model.

Linear mixed models

The **linear mixed model** does something related to the above by-subject fits, but with some crucial twists, as we see below.

The variance associated with subject intercepts (and slopes) is estimated, and from that variance the intercepts (and slopes) for each subject can be predicted.

As we will see the same can be done with the item intercept and slope.

In the model shown in the next slide, we start by introducing random intercepts for subjects only, with the statement (1|subj)

Linear mixed models

```
m0.lmer<-lmer(rt~noise+(1|subj),noisedeg)
```

Abbreviated output:

```
Random effects:
 Groups   Name        Variance Std.Dev.
 subj     (Intercept) 2491     49.91
 Residual             8876     94.21
Number of obs: 40, groups:  subj, 10

Fixed effects:
            Estimate Std. Error t value
(Intercept)   486.00      26.32  18.463
noisenoise     90.00      29.79   3.021
```

Linear mixed models

One thing to notice is that the coefficients of the fixed effects of the above model are identical to those in the linear model m0 above.
The **predicted** (not estimated!) varying intercepts for each subject can be viewed by typing:

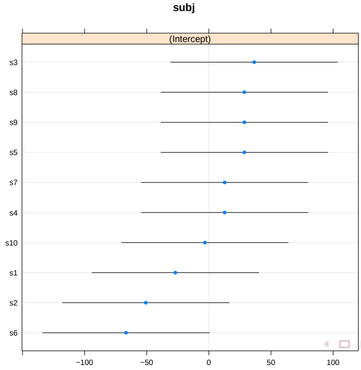
Linear mixed models

```
ranef(m0.lmer)

## $subj
##     (Intercept)
## s1   -26.972985
## s10   -3.173292
## s2   -50.772677
## s3    36.492862
## s4    12.693169
## s5    28.559631
## s6   -66.639139
## s7    12.693169
## s8    28.559631
## s9    28.559631
```

Linear mixed models

Or you can display them graphically.

```
print(dotplot(ranef(m0.lmer,condVar=TRUE)))

## $subj
```



Linear mixed models

The model m0.lmer above prints out the following type of linear model:

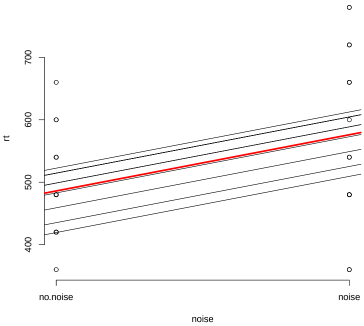
$$Y_{ijk} = \beta_j + b_i + \epsilon_{ijk} \tag{1}$$

$i = 1, \dots, 10$ is subject id, $j = 1, 2$ is the factor level (no noise or noise), k is the number of replicates (here 2).
$b_i \sim N(0, \sigma_b^2), \epsilon_{ijk} \sim N(0, \sigma^2)$.
It's just like our linear model except that there are different *predicted* (cf. the lmlist function above, where they are *estimated* for each subject) intercepts b_i for each subject.

Linear mixed models

Note that these b_i are assumed by lmer to come from a normal distribution centered around 0; see Gelman and Hill 2007 for more.
The ordinary linear model m0 has one intercept β_0 for all subjects, whereas the linear mixed model with varying intercepts m0.lmer has a different intercept $(\beta_0 + b_i)$ for each subject.
We can visualize these different intercepts for each subject as shown below.

Linear mixed models



Linear mixed models

Note that, unlike the figure associated with the lmlist.fm1 model above, which also involves fitting separate models for each subject, the model m0.lmer assumes different intercepts for each subject **but the same slope**.

We can have lmer fit different intercepts AND slopes for each subject.

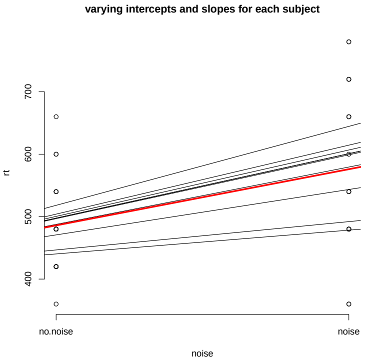
Linear mixed models

```
m1.lmer<-lmer(rt~noise+(1+noise|subj),noisedeg)
```

```
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 subj     (Intercept) 1093     33.05
          noisenoise  1408     37.52    1.00
 Residual             8359     91.43
Number of obs: 40, groups:  subj, 10

Fixed effects:
            Estimate Std. Error t value
(Intercept)   486.00      22.96   21.17
noisenoise     90.00      31.25    2.88
```

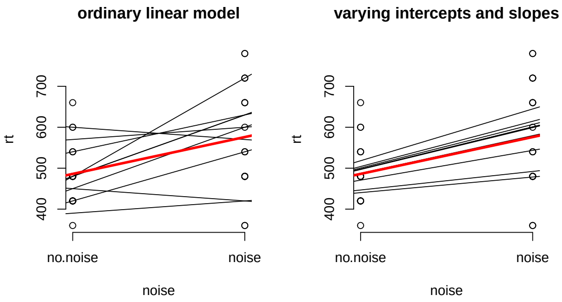
Linear mixed models

These fits for each subject are visualized below (the red line shows the model with a single intercept and slope, i.e., our old model m0):



Linear mixed models

Compare this model with the lmlist.fm1 model we fitted earlier:



Linear mixed models

- The above graphic shows some crucial difference between the lmlist model and the lmer model.
- Note that the fitted line for each subject in the lmer model is much closer to the m0 model's fitted (red) line.
- This is because lmlist uses each subject's data separately (resulting in possibly wildly different models, depending on the variability between subjects), whereas lmer "borrows strength from the mean" and pushes (or "shrinks") the estimated intercepts and slopes of each subject closer to the mean intercepts and slopes (the model m0's intercepts and slopes).

Linear mixed models

- Because it shrinks the coefficients towards the means, this is called shrinkage. This is particularly useful when several data points are missing in a particular condition for a particular subject: estimating coefficients using lmList would lead to very poor estimates for that subject; by contrast, lmer assumes that the estimates for such a subject are not reliable and therefore shrinks that subject's estimate to the mean values.

Linear mixed models

To see an example of shrinkage, we will consider the case where we remove three of the data points from subject s8, resulting in exaggeratedly high means for that subject.

We **are now using the full noisedeg dataset (this is a 2x3 design, with three levels of degree, 0, 4, 8).**

Subject 8 (s8) has only three data points, not six like other subjects (we took out three of s8's low measures). This skews the subject's estimates for intercept and slope in the lmlist model fit.

Let's confirm that the new data frame has extreme means for s8.

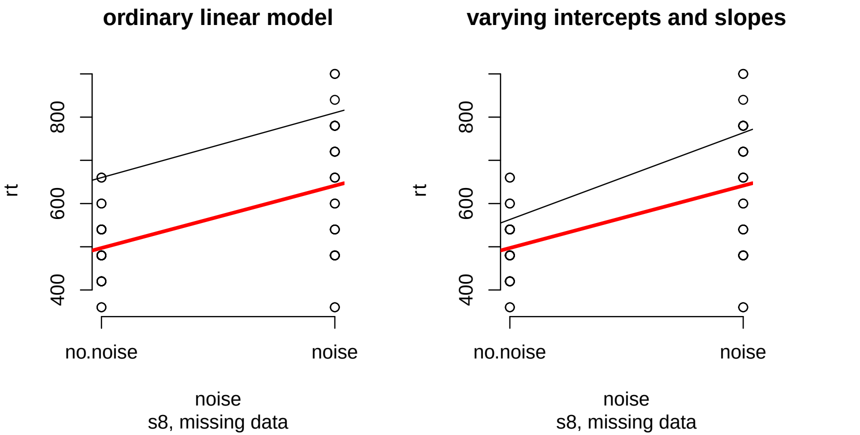
Linear mixed models

```
##      no.noise noise
## s1        440   620
## s10       480   660
## s2        460   480
## s3        500   740
## s4        500   720
## s5        580   620
## s6        380   460
## s7        520   700
## s8        660   810
## s9        560   660
```

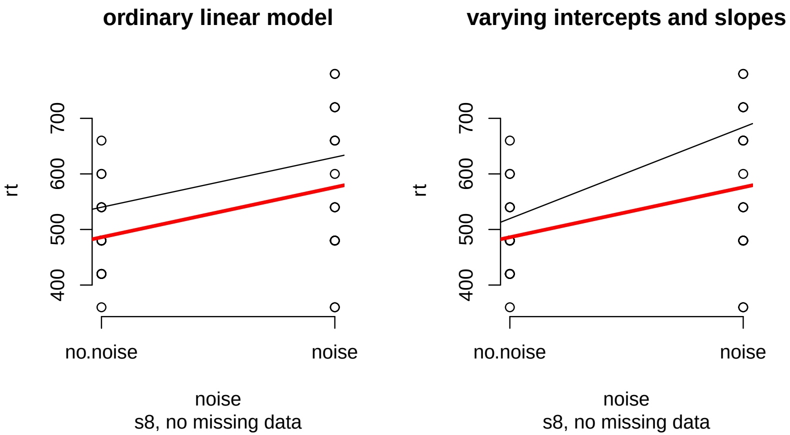
Linear mixed models

If we now fit the lmlist model and the linear mixed model and plot the model for s8, we find that the lmlist model indeed estimates **pretty extreme intercepts for s8**.
But the linear mixed model predicts an intercept that's **much closer to the mean** (the red line).

Linear mixed models



Linear mixed models



Linear mixed models

One crucial difference between the lmlist model and the lmer model is that **lmList estimates the parameters for each subject separately**.
By contrast, **lmer estimates the variance associated with subjects' intercepts** (and slopes, if you specify in the model that one should do that) and then *predicts* each subjects intercepts and slopes based on that variance.