Inference in mixed models in R - beyond the usual asymptotic likelihood ratio test

Søren Højsgaard[1]
Ulrich Halekoh[2]

June 12, 2018

[1] University of Aalborg, Denmark
[2] University of Southern Denmark, Denmark

Example: Double registration in labs

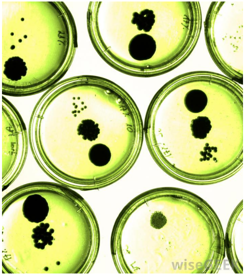


Figure 1

Clustered data:

- Compare two groups (treatment with a control);
- M units (petri plates, persons, animals. . .) per group;
- Each unit is measured R times. Measurements on same unit are positively correlated.

Simulated data: $N = 3$ subjects per group, $R = 2$ replicated measurements per subject.

```
dub
```

```
##      y1 y2  grp  subj
## 1  1.70  0 ctrl subj1
## 2  2.01  0 ctrl subj1
## 3  0.65  0 ctrl subj2
## 4  1.39  2 ctrl subj2
## 5  0.31  1 ctrl subj3
## 6  0.94  0 ctrl subj3
## 7  0.55  0 trt1 subj4
## 8  1.20  2 trt1 subj4
## 9  4.49  4 trt1 subj5
## 10 4.53  5 trt1 subj5
## 11 3.94  2 trt1 subj6
## 12 4.02  0 trt1 subj6
```

Hvis data er korrelerede må man enten kan man vælge forskellige muligheder

- Tage højde for afhængigheden ved at lade den indgå i sin model
- Forsøge, med et gammelt dansk ordvalg, at “komme om ved” afhængigheden.
- Ignorere afhængigheden.

De første to muligheder er helt OK; den sidste er snyd - og det kan gå HELT GALT.

Hvis der er en korrelation, så er den oftest positiv. Problemet med at ignorere problemet er kort fortalt, at man kommer til at “lade som om” der er mere information i data end der i virkeligheden er. Det fører til

- standardafvigelser på parameterestimater bliver for små
- p-værdier bliver for små
- effekter kommer til at se stærkere ud end de i virkeligheden er.

Problem/issues: If we ignore clustering/positive correlation:

- pretending to have more information than we have
- standard errors of estimates become too small
- p values become too small
- effects appear stronger than they really are.

Notice:

- Measuring the same unit many many times will make the dataset larger, but will not really add many more chunks of information (depending on the size of the within-subject correlation, of course).
- Instead, more units are needed.

```r
lg1 <- lm(y1 ~ grp, data=dub)
lg1 %>% summary %>% coef %>% as.data.frame -> tb1
tb1$"Pr(>|X^2|)" = 1 - pchisq(tb1[,3]^2, df=1)
tb1
```

```
##             Estimate Std. Error t value Pr(>|t|) Pr(>|X^2|)
## (Intercept)    1.167     0.5437   2.146  0.05747    0.03189
## grptrt1        1.955     0.7689   2.543  0.02923    0.01100
```

Notice: the *t*-test “accounts for” the uncertainty in the estimate of the standard error.

Alternative: Analyze average

```r
duba <- aggregate(y1 ~ grp + subj, FUN=mean, data=dub)
lm(y1 ~ grp, data=duba) %>% summary %>% coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.167     0.8416   1.386   0.2380
## grptrt1        1.955     1.1903   1.642   0.1758
```

- Works fine (gives the correct test) in (nearly) balanced cases.
- Does not provide estimate of between and within subject variation (not necessarily severe problem here).
- Analyzing-the-average is often not a feasible strategy.

Alternative: Random effects model

```
lg2 <- lmer(y1 ~ grp + (1|subj), data=dub)
tidy(lg2)
```

```
## Warning in bind_rows_(x, .id): binding factor and character vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## # A tibble: 4 x 5
##   term                      estimate std.error statistic group
##   <chr>                        <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)                  1.17      0.842      1.39 fixed
## 2 grptrt1                      1.96      1.19       1.64 fixed
## 3 sd_(Intercept).subj          1.44     NA         NA    subj
## 4 sd_Observation.Residual      0.350    NA         NA    Residual
```

```
sm2 <- update(lg2, .~. -grp)
as.data.frame(anova(lg2, sm2))
```

```
##     Df   AIC   BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## sm2  3 36.80 38.26 -15.40    30.80    NA     NA         NA
## lg2  4 35.71 37.65 -13.86    27.71 3.093      1    0.07864
```

Notice: Test is based in the χ^2 distribution (i.e. that the variance is

Alternatives in the pbkrtest package:

```r
KRmodcomp(lg2, sm2)
```

```
## F-test with Kenward-Roger approximation; time: 0.16 sec
## large : y1 ~ grp + (1 | subj)
## small : y1 ~ (1 | subj)
##       stat ndf ddf F.scaling p.value
## Ftest  2.7 1.0 4.0         1    0.18
```

```r
PBmodcomp(lg2, sm2)
```

```
## Bootstrap test; time: 11.64 sec;samples: 1000; extremes: 180;
## large : y1 ~ grp + (1 | subj)
## small : y1 ~ (1 | subj)
##        stat df p.value
## LRT     2.9  1   0.089
## PBtest  2.9      0.181
```

Notice: Same *p*-value as when analyzing average.

The Kenward–Roger approach

The Kenward–Roger modification of the F–statistic

For multivariate normal data

$$Y_{n\times 1} \sim N(X_{n\times p}\beta_{p\times 1}, \Sigma)$$

we consider the test of the hypothesis

$$L_{d\times p}(\beta - \beta_0) = 0$$

With $\hat{\beta} \sim N_d(\beta, \Phi)$, a Wald statistic is

$$W = [L(\hat{\beta} - \beta_0)]^\top [L\Phi L^\top]^{-1} [L(\hat{\beta} - \beta_0)]$$

which is asymptotically $W \sim \chi^2_d$ under the null hypothesis.

A scaled version of W is

$$F = \frac{1}{d} W$$

- Asymptotically $F \sim \frac{1}{d}\chi^2_d$ under the null hypothesis
- Think of as the limiting distribution of an $F_{d,m}$–distribution as $m \to \infty$
- To account for the fact that $\Phi = \mathbb{V}ar(\hat{\beta})$ is estimated from data, we must come up with a better estimate of the denominator degrees of freedom m (better than $m = \infty$).
- That was what Kenward and Roger worked on. . .

The linear hypothesis $L\beta = \beta_0$ can be tested via the Wald-type statistic

$$F = \frac{1}{r}(\hat{\beta} - \beta_0)^\top L^\top (L^\top \Phi(\hat{\sigma}) L)^{-1} L(\hat{\beta} - \beta_0)$$

- $\Phi(\sigma) = (X^\top \Sigma(\sigma) X)^{-1} \approx \mathbb{C}ov(\hat{\beta})$, $\hat{\beta}$ REML estimate of β
- $\hat{\sigma}$: vector of REML estimates of the elements of $\Sigma = \mathbb{V}ar(Y)$

Kenward and Roger (1997)

- replaced Φ by an improved small sample approximation Φ_A
- scaled F by a factor λ
- determined denominator degrees of freedom m by matching moments of F/λ with an $F_{d,m}$ distribution.

Shortcommings of Kenward-Roger

- The Kenward–Roger approach is no panacea.
- In the computations of the degrees of freedom we need to compute
$$G_j \Sigma^{-1} G_j$$
where $\Sigma = \sum_i \sigma_i G_i$. Can be space and time consuming!
- An alternative is a Sattherthwaite–kind approximation which is faster to compute. Will come out in next release of `pbkrtest` (code not tested yet). Way faster...
- What to do with generalized linear mixed models – or even with generalized linear models.
- `pbkrtest` also provides the parametric bootstrap *p*-value. Computationally somewhat demanding, but can be parallelized.

Parametric bootstrap

We have two competing models; a large model $f_1(y;\theta)$ and a null model $f_0(y;\theta_0)$; the null model is a submodel of the large model.

- The p value for a composite hypothesis is

  $$p = \sup_{\theta \in \Theta_0} Pr_\theta(T \geq t_{obs})$$

  where the sup is taken under the hypothesis.
- We can (usually) not evaluate the sup in practice, so instead we do:

  $$p^{PB} = Pr_{\hat{\theta}}(T \geq t_{obs})$$
- In practice we approximate p^{PB} as
  - Draw B parametric bootstrap samples $t^1, \dots, t^B$ under the fitted null model $\hat{\theta}_0$.
  - Fit the large and the null model to each of these datasets;
  - Calculate the LR-test statistic for each simulated data; this gives reference distribution.
  - Calculate how extreme the observed statistic is.

```r
lg2 <- update(lg2, REML=FALSE)
sm2 <- update(sm2, REML=FALSE)
# Observed test statistic:
t.obs <- 2 * (logLik(lg2) - logLik(sm2))
t.obs
```

```
## 'log Lik.' 3.093 (df=4)
```

```r
# Reference distribution
set.seed(121315)
t.sim <- PBrefdist(lg2, sm2, nsim=2000)
# p-value
head(t.sim)
```

```
## [1] 0.6139 1.6109 2.8410 0.2371 2.5034 1.7788
```

```r
sum(t.sim >= t.obs) / length(t.sim)
```

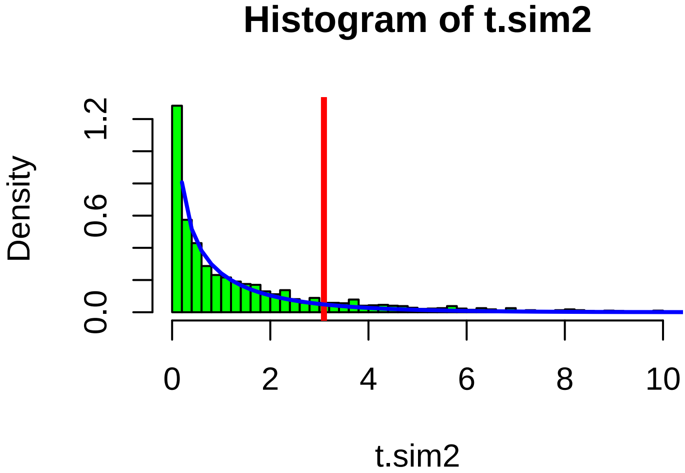
```
## [1] 0.175
```

```r
# compare with X^2 dist
1 - pchisq(t.obs, df=1)
```

```
## 'log Lik.' 0.07864 (df=4)
```

Interesting to overlay limiting χ_1^2 distribution and simulated reference distribution.

Bootstrap reference distribution has heavier tail giving larger *p*-value.



Speedup I: Sequential *p*-value

Instead of simulating a fixed number of values $t^1, \dots, t^B$ for determining the reference distribution used for finding p^{PB} we may instead introduce a stopping rule saying *simulate until we have found, say $h = 20$ values t^j larger than t_{obs}*. If J simulations are made then the reported p–value is h/J.

```r
spb <- seqPBmodcomp(lg2, sm2)
spb
```

```
## Bootstrap test; time: 2.82 sec;samples: 200; extremes: 38;
## large : y1 ~ grp + (1 | subj)
## small : y1 ~ (1 | subj)
##        stat df p.value
## LRT    3.09  1   0.079
## PBtest 3.09      0.194
```

Speedup II: Parallel computations

Parametric bootstrap is computationally demanding, but multiple cores can be exploited. Done by default on linux / mac platforms.

```
PBmodcomp(lg2, sm2) # Default: Use all cores (4 on my computer)
```

```
## Bootstrap test; time: 10.98 sec;samples: 1000; extremes: 171;
## large : y1 ~ grp + (1 | subj)
## small : y1 ~ (1 | subj)
##        stat df p.value
## LRT    3.09  1   0.079
## PBtest 3.09      0.172
```

```
PBmodcomp(lg2, sm2, cl=1) # Use one core
```

```
## Bootstrap test; time: 19.41 sec;samples: 1000; extremes: 179;
## large : y1 ~ grp + (1 | subj)
## small : y1 ~ (1 | subj)
##        stat df p.value
## LRT    3.09  1   0.079
## PBtest 3.09      0.180
```

On windows (in fact, work on all platforms):

```r
set.seed(121315)
library(parallel)
nc <- detectCores(); nc
clus <- makeCluster(rep("localhost", nc))
PBmodcomp(lg2, sm2, cl=clus)
```

Speedup III: Parametric form of reference distribution:

Estimating tail–probabilities will require more samples than estimating the mean (and variance) of the reference distribution.

Suggests to approximate simulated reference distribution with a known distribution so that fewer samples will suffice:

```
pb1 <- PBmodcomp(lg2, sm2, nsim=1000)
pb2 <- PBmodcomp(lg2, sm2, nsim=100)
summary(pb1) %>% as.data.frame
```

```
##             stat df   ddf p.value
## LRT        3.093  1    NA 0.07864
## PBtest     3.093 NA    NA 0.17782
## Gamma      3.093 NA    NA 0.18370
## Bartlett 1.767  1    NA 0.18370
## F          3.093  1 4.667 0.14313
```

```
summary(pb2) %>% as.data.frame
```

```
##             stat df   ddf p.value
## LRT        3.093  1    NA 0.07864
## PBtest     3.093 NA    NA 0.14851
## Gamma      3.093 NA    NA 0.13455
## Bartlett 2.181  1    NA 0.13974
## F          3.093  1 6.783 0.12341
```

Why use parametric bootstrap

- Applies generally; in `pbkrtest` implemented for e.g. generalized linear mixed models (hwere random effects are on the linear predictor scale).
- Kenward-Roger does not readily scale to larger problems because of the computation of

  $$G_j \Sigma^{-1} G_j$$

  where $\Sigma = \sum_i \sigma_i G_i$. Can be space and time consuming!
- For example, in random regression models with few relatively long time series. In this case simulation is faster.

Simulation study

```
dub
```

```
##       y1 y2  grp  subj
## 1  1.70  0 ctrl subj1
## 2  2.01  0 ctrl subj1
## 3  0.65  0 ctrl subj2
## 4  1.39  2 ctrl subj2
## 5  0.31  1 ctrl subj3
## 6  0.94  0 ctrl subj3
## 7  0.55  0 trt1 subj4
## 8  1.20  2 trt1 subj4
## 9  4.49  4 trt1 subj5
## 10 4.53  5 trt1 subj5
## 11 3.94  2 trt1 subj6
## 12 4.02  0 trt1 subj6
```

- Task: Test the hypothesis that there is no effect of treatment. How good are the various tests?
- Simulate data 1000 times with divine insight: there is no effect of treatment.
- Test the hypothesis e.g. at level 5%. If test has correct nominal level we shall reject about 50 times.
- If hypothesis is rejected e.g. 100 times then p values are anti-conservative: Effects appear more significant than the really are. That is we draw “too strong” conclusions.

	0.010	0.050	0.100
lm+X2	0.178	0.282	0.342
lm+F	0.110	0.240	0.322
mixed+X2	0.044	0.152	0.240
mixed+F-KR	0.012	0.044	0.114
mixed+PB	0.008	0.052	0.108

Motivation: Sugar beets - A split–plot experiment

- Model how sugar percentage in sugar beets depends on harvest time and sowing time.
- Five sowing times (s) and two harvesting times (h).
- Experiment was laid out in three blocks (b).

Experimental plan for sugar beets experiment

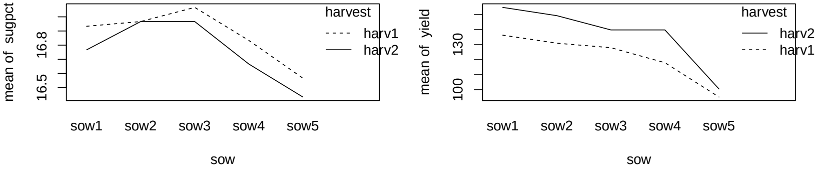
```
# Plot allocation:
#       |  Block 1       |  Block 2       |  Block 3       |
#       +----------------|----------------|----------------+
# Plot  | h1 h1 h1 h1 h1 | h2 h2 h2 h2 h2 | h1 h1 h1 h1 h1 | Harvest time
# 1-15  | s3 s4 s5 s2 s1 | s3 s2 s4 s5 s1 | s5 s2 s3 s4 s1 | Sowing time
#       |----------------|----------------|----------------|
# Plot  | h2 h2 h2 h2 h2 | h1 h1 h1 h1 h1 | h2 h2 h2 h2 h2 | Harvest time
# 16-30 | s2 s1 s5 s4 s3 | s4 s1 s3 s2 s5 | s1 s4 s3 s2 s5 | Sowing time
#       +----------------|----------------|----------------+
```

beets data

```
data(beets, package='pbkrtest')
head(beets)
```

```
##   harvest  block  sow yield sugpct
## 1   harv1 block1 sow3 128.0   17.1
## 2   harv1 block1 sow4 118.0   16.9
## 3   harv1 block1 sow5  95.0   16.6
## 4   harv1 block1 sow2 131.0   17.0
## 5   harv1 block1 sow1 136.5   17.0
## 6   harv2 block2 sow3 136.5   17.0
```

```
par(mfrow=c(1,2))
with(beets, interaction.plot(sow, harvest, sugpct))
with(beets, interaction.plot(sow, harvest, yield))
```



- For simplicity assume no interaction between sowing and harvesting times.
- A typical model for such an experiment would be:

$$y_{hbs} = \mu + \alpha_h + \beta_b + \gamma_s + U_{hb} + \epsilon_{hbs}, \tag{1}$$

  where $U_{hb} \sim N(0, \omega^2)$ and $\epsilon_{hbs} \sim N(0, \sigma^2)$.
- Notice that U_{hb} describes the random variation between whole–plots (within blocks).

Using `lmer()` from lme4 we can test for no effect of sowing and harvest time as:

```
beet.lg <- lmer(sugpct ~ block + sow + harvest +
                      (1 | block:harvest), data=beets, REML=FALSE)
beet.noh <- update(beet.lg, .~. - harvest)
beet.nos  <- update(beet.lg, .~. - sow)
```

Both factors appear highly significant

```
anova(beet.lg, beet.noh)  %>% as.data.frame
```

```
##          Df    AIC    BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## beet.noh  9 -69.08 -56.47  43.54   -87.08    NA     NA         NA
## beet.lg  10 -80.00 -65.99  50.00  -100.00 12.91      1  0.0003261
```

```
anova(beet.lg, beet.nos)  %>% as.data.frame
```

```
##          Df     AIC     BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## beet.nos  6  -2.795   5.612  7.398    -14.8    NA     NA         NA
## beet.lg  10 -79.998 -65.986 49.999   -100.0  85.2      4  1.374e-17
```

However, the LRT based *p*–values are anti–conservative: the effect of harvest appears stronger than it is.

As the design is balanced we may make F–tests for each of the effects as:

```
beets$bh <- with(beets, interaction(block, harvest))
summary(aov(sugpct ~ block + sow + harvest +
                Error(bh), data=beets))
```

```
##
## Error: bh
##           Df Sum Sq Mean Sq F value Pr(>F)
## block      2 0.0327  0.0163    2.58   0.28
## harvest    1 0.0963  0.0963   15.21   0.06
## Residuals  2 0.0127  0.0063
##
## Error: Within
##           Df Sum Sq Mean Sq F value  Pr(>F)
## sow        4   1.01  0.2525     101 5.7e-13
## Residuals 20   0.05  0.0025
```

Notice: the F–statistics are $F_{1,2}$ for harvest time and $F_{4,20}$ for sowing time.

```r
set.seed("260618")
KRmodcomp(beet.lg, beet.noh)
```

```
## F-test with Kenward-Roger approximation; time: 0.34 sec
## large : sugpct ~ block + sow + harvest + (1 | block:harvest)
## small : sugpct ~ block + sow + (1 | block:harvest)
##       stat  ndf  ddf F.scaling p.value
## Ftest 15.2  1.0  2.0          1    0.06
```

```r
PBmodcomp(beet.lg, beet.noh)
```

```
## Bootstrap test; time: 12.56 sec;samples: 1000; extremes: 32;
## large : sugpct ~ block + sow + harvest + (1 | block:harvest)
## small : sugpct ~ block + sow + (1 | block:harvest)
##         stat df p.value
## LRT    12.9  1 0.00033
## PBtest 12.9    0.03297
```

```r
seqPBmodcomp(beet.lg, beet.noh)
```

```
## Bootstrap test; time: 12.59 sec;samples: 800; extremes: 28;
## large : sugpct ~ block + sow + harvest + (1 | block:harvest)
## small : sugpct ~ block + sow + (1 | block:harvest)
##         stat df p.value
## LRT    12.9  1 0.00033
## PBtest 12.9    0.03620
```

Final remarks

- Satterthwaite approximation of degrees of freedom on its way in `pbkrtest`. Faster to compute than Kenward-Roger scales to larger problems.
- `pbkrtest` available on CRAN `https://cran.r-project.org/package=pbkrtest`
- devel version on github: `devtools::install_github(hojsgaard/pbkrtest)`
- `pbkrtest` described in Ulrich Halekoh and SH (2014) A Kenward-Roger Approximation and Parametric Bootstrap Methods for Tests in Linear Mixed Models The R Package pbkrtest; Journal of Statistical Software, Vol 59.

Thanks for your attention!