Simulation to understand mixed effects models

SST 2014 Workshops

Dave Kleinschmidt

Brain and Cognitive Sciences
University of Rochester

December 2, 2014

Virtual reality

- Mixed models can be complicated and behave in non-intuitive ways
- Can make it hard to interpret results when analyzing your real data
- How do you know what the effect of your modeling decisions are?
- *Forget about real data*
- Use *simulated* data to probe model behavior.

Code: simulate data

```
library(plyr)
library(mvtnorm)
library(lme4)

make.data.generator <- function(true.effects=c(0,0),
                                resid.var=1,
                                ranef.covar=diag(c(1,1)),
                                n.subj=24,
                                n.obs=24
                                )
{
  # create design matrix for our made up experiment
  data.str <- data.frame(freq=factor(c(rep('high', n.obs/2), rep('low', n.obs/2))))
  contrasts(data.str$freq) <- contr.sum(2)
  model.mat <- model.matrix(~ 1 + freq, data.str)

  generate.data <- function() {
    # sample data set under mixed effects model with random slope/intercepts
    simulated.data <- rdply(n.subj, {
      beta <- t(rmvnorm(n=1, sigma=ranef.covar)) + true.effects
      expected.RT <- model.mat %*% beta
      epsilon <- rnorm(n=length(expected.RT), mean=0, sd=sqrt(resid.var))
      data.frame(data.str,
                 RT=expected.RT + epsilon)
    })
    names(simulated.data)[1] <- 'subject'
    simulated.data
  }
}
```

Code: fit model


```
fit.models <- function(simulated.data) {
  # fit models and extract coefs
  lm.coefs <- coefficients(summary(lm(RT ~ 1+freq, simulated.data)))[, 1:3]
  rand.int.coefs <- coefficients(summary(lmer(RT ~ 1+freq + (1|subject), simulated.data)))
  rand.slope.coefs <- coefficients(summary(lmer(RT ~ 1+freq + (1+freq|subject), simulated.data)))
  # format output all pretty
  rbind(data.frame(model='lm', predictor=rownames(lm.coefs), lm.coefs),
        data.frame(model='rand.int', predictor=rownames(rand.int.coefs), rand.int.coefs),
        data.frame(model='rand.slope', predictor=rownames(rand.slope.coefs), rand.slope.coefs))
}
```

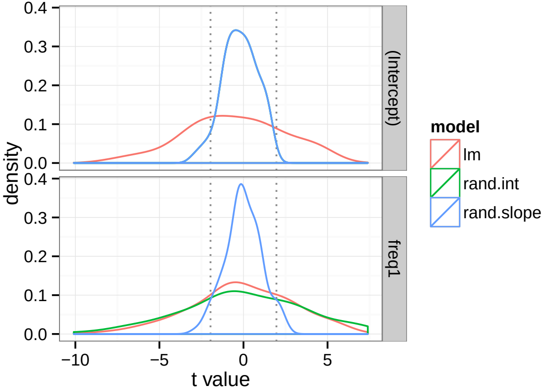
Generate data and fit models


```
gen.dat <- make.data.generator(true.effects=c(0,0), n.subj=24, n.obs=24)
simulations <- rdply(.n=100,
                     fit.models(gen.dat()))
```

```
head(simulations)

##   .n      model   predictor   Estimate Std..Error    t.value
## 1  1         lm (Intercept) -0.1039682 0.07204041 -1.4431925
## 2  1         lm       freq1  0.2368729 0.07204041  3.2880553
## 3  1   rand.int (Intercept) -0.1039682 0.22225682 -0.4677840
## 4  1   rand.int       freq1  0.2368729 0.05783136  4.0959241
## 5  1 rand.slope (Intercept) -0.1039682 0.22225685 -0.4677839
## 6  1 rand.slope       freq1  0.2368729 0.20278075  1.1681230

daply(simulations, .(model, predictor), function(df) type1err=mean(abs(df$t.value)>1.96))

##             predictor
## model        (Intercept) freq1
##   lm                0.53  0.51
##   rand.int          0.05  0.56
##   rand.slope        0.05  0.09
```

Visualize

```r
# use reshape2::melt to get the data into a more convenient format (see next section)
ggplot(simulations, aes(x=t.value, color=model)) +
  geom_vline(xintercept=c(-1.96, 1.96), color='#888888', linetype=3) +
  scale_x_continuous('t value') +
  geom_density() +
  facet_grid(predictor~.)
```



Power vs. false-positive simulations

- Power is impacted by *sample size*.
- True both of observation sample size and number of subjects

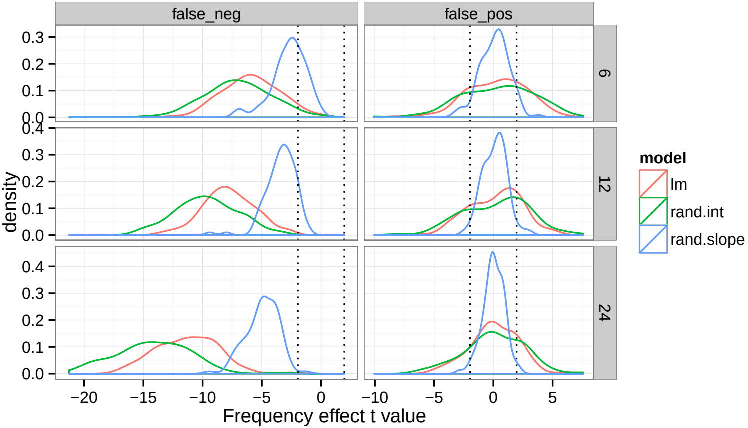
Simulation code

```
## parameters for lex dec, log rts, categorical frequency (high/low)
intercept <- 6.4                        # mean log-RT
freq_slope <- -0.06                     # high vs. low frequency
bysub_sd <- c(0.15, 0.06)              # random effect sd
resid_sd <- 0.18                        # residual sd

n_subj <- 21                            # number of subjects in original data
n_trials <- 80                          # number of trials each subject did

## parameters to vary in simulation: number of subjects
sim.params <- expand.grid(n.subj=c(6, 12, 24))

## data generator factory functions for false negatives and false positives
falseneg.factory <- function(...) {
  make.data.generator(true.effects = c(intercept, freq_slope),
                      resid.var = resid_sd^2,
                      ranef.covar = diag(bysub_sd^2),
                      n.obs = n_trials,
                      ...)
}

falsepos.factory <- function(...) {
  make.data.generator(true.effects = c(0,0),
                      resid.var = resid_sd^2,
                      ranef.covar = diag(bysub_sd^2),
                      n.obs = n_trials,
                      ...)
}
```

Run simulations

```
## run simulations for effect present (falseneg) and absent (falsepos)
sims.falseneg <- mdply(sim.params,
                       function(...) {
                         gen.dat <- falseneg.factory(...)
                         rdply(.n=100, fit.models(gen.dat()))
                       })

sims.falsepos <- mdply(sim.params,
                       function(...) {
                         gen.dat <- falsepos.factory(...)
                         rdply(.n=100, fit.models(gen.dat()))
                       })

sims <- rbind(data.frame(sims.falseneg, type='false_neg'),
              data.frame(sims.falsepos, type='false_pos'))
```

Visualize: t-value distributions



Visualize: power vs. false-positives