R Code from
Mahler's Guide to

Linear Mixed Models

CAS Exam MAS-2

prepared by
Howard C. Mahler, FCAS

Howard Mahler
hmahler@mac.com
www.howardmahler.com/Teaching

For my Weight Loss Example (in my Section 2):

```
install.packages("nlme") # you only have to install a package once
library(nlme)
individual=c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6)
time=c(0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3)
weight=c(154,150,144,142,187,180,175,169,149,147,140,134,222,214,206,196,176,167,164,
                160,130,126,120,118)

simplemodel<-lme(weight~time, random = ~1 | individual, method = "ML")
summary(simplemodel)
random.effects(simplemodel)

# Fitting instead via Restricted Maximum Likelihood
simplemodel2<-lme(weight~time, random = ~1 | individual, method = "REML")
summary(simplemodel2)
random.effects(simplemodel2)
```

For my Claim Counts Example (in my Section 2):

```
install.packages("nlme") # you only have to install a package once
library(nlme)

prior=c(0, 0, 1, 0, 1, 0, 1, 1, 2, 3, 4, 2, 1, 3, 3, 0, 1, 4, 5, 2, 1, 0, 2, 2, 2, 3, 1, 2, 0, 1, 0, 1, 3, 1, 0,
        5, 2, 2, 6, 2, 0, 0, 1, 1, 1, 1, 2, 3, 1, 2, 4, 2, 3, 4, 0, 2, 4, 2, 1, 8, 0, 0, 2, 1, 5, 2, 3, 4, 5, 4,
        0, 2, 1, 0, 3, 2, 1, 4, 5, 3,2, 2, 1, 5, 5, 4, 1, 1, 5, 2, 2, 4, 4, 2, 5, 3, 6, 0, 5, 3, 1, 2, 0, 0, 4, 1,
        6, 1, 2, 2, 4, 2, 1, 1, 0, 6, 2, 1, 4, 2)

post=c(0, 0, 1, 0, 1, 3, 0, 0, 0, 3, 0, 0, 2, 3, 2, 2, 1, 4, 1, 3, 1, 3, 0, 1, 0, 0, 4, 0, 1, 3, 4, 0, 3, 0, 3,
        2, 2, 2, 1, 5, 3, 2, 1, 1, 2, 0, 3, 1, 2, 2, 7, 1, 1, 4, 4, 3, 2, 2, 6, 6, 1, 1, 1, 3, 0, 1, 0, 3, 3, 1,
        2, 2, 3, 4, 5, 2, 5, 5, 3, 4, 1, 2, 1, 4, 3, 2, 2, 1, 2, 1, 6, 4, 0, 2, 6, 2, 5, 3, 3, 3, 4, 2, 1, 3, 6,
        3, 0, 1, 5, 2, 5, 3, 2, 2, 3, 3, 5, 2, 7, 4)

class=c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
        3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,
        5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6)

simplemodelcounts<-lme(post~prior, random = ~1 | class, method = "ML")
summary(simplemodelcounts)
random.effects(simplemodelcounts)

# Fitting instead via Restricted Maximum Likelihood
simplemodelcounts2<-lme(post~prior, random = ~1 | class, method = "REML")
summary(simplemodelcounts2)
random.effects(simplemodelcounts2)
```

For my Grain Yield Example (in my Section 2):

```
install.packages("lme4") # you only have to install a package once
library(lme4)

seed=c(0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1)

plot=c(1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4)

year=c(1,1,1,1,2,2,2,2,3,3,3,3,1,1,1,1,2,2,2,2,3,3,3,3)

yield=c(920,1084,1195,954,722,691,690,835,808,972,1002,1295,
        1232,1168,1017,1220,933,1004,907,989,762,1029,1086,1209)

simplemodelyields<-lmer(yield~seed + (1 | plot) + (1 | year), REML = F)
summary(simplemodelyields)
ranef(simplemodelyields)

# Fitting instead via Restricted Maximum Likelihood
simplemodelyields2<-lmer(yield~seed + (1 | plot) + (1 | year), REML = T)
summary(simplemodelyields2)
ranef(simplemodelyields2)
```

For the Sleep Study Example (in my Section 2):

```
install.packages("lme4") # you only have to install a package once
library(lme4)

attach(sleepstudy)
sleepstudy[1:13,]
require(lattice)
xyplot(Reaction ~ Days | Subject, sleepstudy, type = c("g","p","r"),
        index = function(x,y) coef(lm(y ~ x))[1],
        xlab = "Days of sleep deprivation",
        ylab = "Average reaction time (ms)", aspect = "xy")

fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy, subset=Days>=2)
summary(fm1)
ranef(fm1)

install.packages("merTools") # you only have to install a package once
library(merTools)
plotREsim(REsim(fm1)[1:18,])
plotREsim(REsim(fm1)[19:36,])

# By default, lme4 assumes that all coefficients associated with the same random-effects term
        are correlated.
# A model with instead independent random effects
fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy, subset=Days>=2)
summary(fm2)
ranef(fm2)
```

For the EM Algorithm Example (in my Section 5):

```
install.packages("mixtools")  # only need to install a package once
library(mixtools)

ESL=c(-0.39,0.12,0.94,1.67,1.76,2.44,3.72,4.28,4.92,5.53,0.06,0.48,1.01,1.68,1.80,3.25,4.12,
        4.60,5.28,6.22)

hist(ESL)

normalmixEM(ESL, lambda=0.5, mu=c(1,4), sigma=c(1,1))
# Algorithm converged in 16 iterations
```

For My Revised Simple Weight Loss Example (in my Section 9):

```
install.packages("nlme") # you only have to install a package once
library(nlme)
individual=c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6)
time=c(0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3)
weight2=c(164,160,154,152,177,170,165,159,169,167,160,154,172,164,156,146,176,167,164,
          160,170,166,160,158)
simplemodel3<-lme(weight2~time, random = ~1 | individual, method = "REML")
summary(simplemodel3)
random.effects(simplemodel3)
predict(simplemodel3)
```

For Simulating the Authors' Example 1 (in mySection 15):

```
install.packages("nlme") # you only have to install a package once
library(nlme)
simweight <- function(intercept, sigma){
        weight<-intercept+rnorm(4,0,sigma)
        return(weight) }
simweights <- function(samples, sigma, sigmalitter, diff){
        weights <- rep(NA, times = samples*8)
        for (i in 1:samples){
        # mean weight for treatment 1 is assumed to be 10
        weights[4*(i-1)+1:4]<-simweight(10+rnorm(1,0,sigmalitter),sigma)
        # mean weight for treatment 2 is assumed to be 10 + diff
        weights[4*samples +4*(i-1)+1:4]<-simweight(10 + diff + rnorm(1,0,sigmalitter),sigma) }
        return(weights) }

createlitter<-function(samples){
        lit<-rep(1,4)
        for (i in 2:(samples*2)){
        lit<-c(lit,rep(i,4))}
        return(lit)}
createtreat<-function(samples){
        t<-c(rep(1,4*samples), rep(2,4*samples))
        return(t)}
testdiff<-function(samples, sigma, sigmalitter, litter, treatment, diff){
        w<-simweights(samples, sigma, sigmalitter, diff)
        pvalue<-anova(lme(w~treatment, random = ~1 | litter, method = "ML"))[2,4]
        return(pvalue)}
testdiffs <- function(trials, samples, sigma, sigmalitter, litter, treatment, diff){
        pvalues <- rep(NA, times = trials)
        for (i in 1:trials){
        pvalues[i]<-testdiff(samples, sigma, sigmalitter, litter, treatment, diff)}
        return(pvalues)}

simpowers <- function(trials, samples, sigma, sigmalitter, diff){
        litter <-createlitter(samples)
        treatment<-createtreat(samples)
        powers<- rep(NA, times = 5)
        pp<-testdiffs(trials, samples, sigma, sigmalitter, litter, treatment, diff)
        powers[1]<-sum(pp<0.1)
        powers[2]<-sum(pp<0.05)
        powers[3]<-sum(pp<0.025)
        powers[4]<-sum(pp<0.01)
        powers[5]<-sum(pp<0.005)
        powers<-powers/trials
        return(powers) }
simpowers(10000, 10, 0.4, 0.3, 0.5)
```

For Simulating the Simple Weight Example (in my Section 15):

```
install.packages("nlme") # you only have to install a package once
library(nlme)

simweight <- function(intercept, slope, sigma){
        weight<-intercept+c(0,1,2,3)*slope+rnorm(4,0,sigma)
        return(weight) }

simweights <- function(samples, Beta0, slope, sigma, sigmaindiv){
        weights <- rep(NA, times = samples*4)
        for (i in 1:samples){
        weights[4*(i-1)+1:4]<-simweight(Beta0+rnorm(1,0,sigmaindiv),slope,sigma) }
        return(weights) }

createindiv<-function(samples){
        ind<-rep(1,4)
        for (i in 2:samples){
        ind<-c(ind,rep(i,4))}
        return(ind)}

testslope<-function(samples, Beta0, slope, sigma, sigmaindiv, individual, time){
        w<-simweights(samples,Beta0, slope, sigma, sigmaindiv)
        pvalue<-anova(lme(w~time, random = ~1 | individual, method = "ML"))[2,4]
        return(pvalue)}

testslopes <- function(trials, samples, Beta0, slope, sigma, sigmaindiv, individual, time){
        pvalues <- rep(NA, times = trials)
        for (i in 1:trials){
        pvalues[i]<-testslope(samples,Beta0, slope, sigma, sigmaindiv, individual, time)}
        return(pvalues)}

simpowers <- function(trials, samples, Beta0, slope, sigma, sigmaindiv){
        individual<-createindiv(samples)
        time<-rep(c(0,1,2,3),samples)
        powers<- rep(NA, times = 5)
        pp<-testslopes(trials,samples,Beta0, slope, sigma, sigmaindiv, individual, time)
        powers[1]<-sum(pp<0.1)
        powers[2]<-sum(pp<0.05)
        powers[3]<-sum(pp<0.025)
        powers[4]<-sum(pp<0.01)
        powers[5]<-sum(pp<0.005)
        powers<-powers/trials
        return(powers) }

simpowers(10000, 6, 160, -1, 2, 10)
```

For Using Simulation to Estimate the Power of a Warranty Example (in my Section 18):

```
install.packages("nlme") # you only have to install a package once
library(nlme)

simlogpay <- function(numclaims) {
        logpay <- rnorm(numclaims, mean=1, sd=1)
        logpay <- c(logpay, rnorm(numclaims, mean=2, sd=2) )
        logpay <- c(logpay, rnorm(numclaims, mean=3, sd=3) )
        logpay <- c(logpay, rnorm(numclaims, mean=4, sd=4) )
        return(logpay) }
simlogpays <- function(nstores, nclaims, sigmastore){
        lp <- rep(NA, times = 4*nstores*nclaims)
        for (i in 1: nstores){
        lp[4*nclaims*(i-1)+ 1:(4*nclaims)]<-simlogpay(nclaims) + rnorm(1, 0, sigmastore) }
        return(lp) }

createstore<-function(nstores, nclaims){
        st<-rep(1,4*nclaims)
        for (i in 2:nstores){
        st<-c(st,rep(i,4*nclaims))}
        return(st)}

createlag<-function(nstores, nclaims){
        lag<-c(rep(1,nclaims), rep(2,nclaims), rep(3,nclaims), rep(4,nclaims))
        for (i in 2:nstores){
        lag<-c(lag, rep(1,nclaims), rep(2,nclaims), rep(3,nclaims), rep(4,nclaims))}
        return(lag)}

testhet<-function(nstores, nclaims, sigmastore, lags, stores){
        logpay<-simlogpays(nstores, nclaims, sigmastore)
        m1<-lme(logpay~lags, random = ~1 | stores, method = "REML")
        # the 2nd model assumes that the variance of residuals is a power function of the lags
        wf<-varPower(form=~lags)
        m2<-lme(logpay~lags, random = ~1 | stores, method = "REML", weights = wf)
        pvalue<-anova(m1,m2)[2,9]
        return(pvalue)}

testhets <- function(trials, nstores, nclaims, sigmastore, lags, stores){
        pvalues <- rep(NA, times = trials)
        for (i in 1:trials){
        pvalues[i]<-testhet(nstores, nclaims, sigmastore, lags, stores)}
        return(pvalues)}
```

```
simpowers <- function(trials, nstores, nclaims, sigmastore){
        lags <-createlag(nstores, nclaims)
        stores<-createstore(nstores, nclaims)
        powers<- rep(NA, times = 5)
        pp<-testhets(trials, nstores, nclaims, sigmastore, lags, stores)
        powers[1]<-sum(pp<0.1)
        powers[2]<-sum(pp<0.05)
        powers[3]<-sum(pp<0.025)
        powers[4]<-sum(pp<0.01)
        powers[5]<-sum(pp<0.005)
        powers<-powers/trials
        return(powers) }

simpowers(10000, 3, 2, 1)
```