MoTrPAC: power calculations for mixed effects datasets with time

David Amar

9/15/2018

The underlying model

We simulate a linear mixed effects models with a random intercept model, where each subject (g) has its own intercept but the subjects share a common transient trend. There are five time points per subject: one pre- (time point zero) and four post-exercise (time points 1-4). The transient reponse peaks at the second post-exercise time point, whereas time points 1 and 3 have 25/% of the peak effect. The last time point has no effect. We use the lme4 and simr packages to calculate the detection power for different parameters. We first show example datasets with different within vs. between subject variability ratios. We fix the between variance to 1 and change the within variance.

```
# Helper functions
# Simulate a linear mixed effects dataset
simulate_data<-function(n_t,sigma_between,sigma_within,effects_vec,
                        n_subjects,effect_size,use_arima=F,arima_rho=0.5){
  intecepts = rnorm(n_subjects,sd = sigma_between)
  y = c()
  g = c()
  tp = c()
  for(j in 1:n_subjects){
    errs = rnorm(n_t,sd=sigma_within)
    if(use_arima){
      errs = arima.sim(list(order = c(1,0,0), ar = arima_rho),
                      n = n_t,sd = sigma_within)
    }
    effs = effect_size * effects_vec
    y = c(y,intecepts[j]+errs+effs)
    g = c(g,rep(j,n_t))
    tp = c(tp,0:(n_t-1))
  }
  d = data.frame(y,g,t=tp)
  return(d)
}
# Plot the trajectories of the subjects
plot_longi<-function(d,...){
  plot(y=d$y,d$t,type="n",ylab="abundance",...)
  points(y=d$y,d$t,type="p",pch=3)
  for(i in unique(d$g)){lines(y=d$y[d$g==i],d$t[d$g==i],type="l",lty=2)}
  lines(lowess(y=d$y,d$t,f=0.1),lwd=3,col="yellow")
}

# Example dataset:
n_t = 5 # one for pre and then four time points
sigma_between = 2 # random effect standard deviation
n_subjects = 50
```

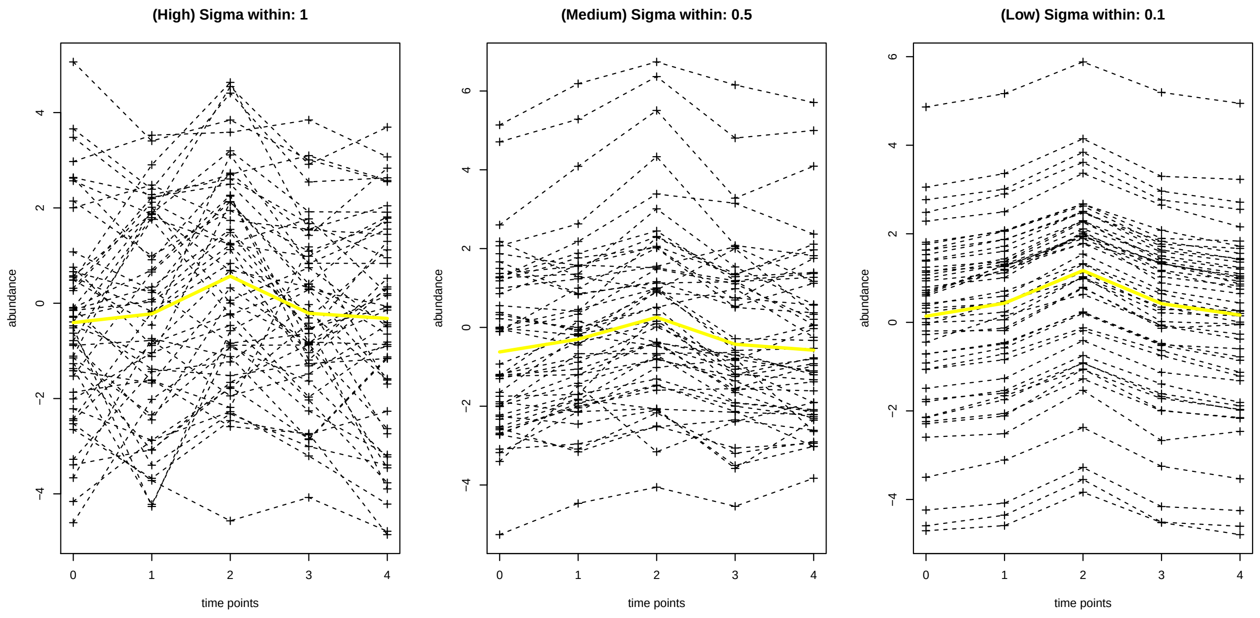

Figure 1: Simulated data examples with different within variance parameters. Greate values lead to a 'noisy' plot. Yellow lines represent the average time reponse.

```
effects_vec = c(0,0.25,1,0.25,0)
effect_size = 1
par(mfrow=c(1,3))
sigma_within = 1
d = simulate_data(n_t,sigma_between,sigma_within ,effects_vec,n_subjects,effect_size)
plot_longi(d,xlab="time points",main="(High) Sigma within: 1")
sigma_within = 0.5
d = simulate_data(n_t,sigma_between,sigma_within ,effects_vec,n_subjects,effect_size)
plot_longi(d,xlab="time points",main="(Medium) Sigma within: 0.5")
sigma_within = 0.1
d = simulate_data(n_t,sigma_between,sigma_within ,effects_vec,n_subjects,effect_size)
plot_longi(d,xlab="time points",main="(Low) Sigma within: 0.1")
```

Power calculation considerations

Obviously, if the time response of an anlyte is transient and involves time points that are not covered in the study then the detection power is zero. Here, we consider power calculations for the example datasets as in Figure 1 but with different within variance and effect size parameters. We also consider two different methods for analyte discovery: (1) a naive LMM that treats time as factors, and (2) fitting linear and cubic orthogonal polynomials. Given our transient response, we expect that the 2-degree polynomial will detect the differential abundance pattern.

```
library(lme4);library(simr);library(gplots)
#' Power analysis that treats time as simple factors
#'
#'  @param d A data frame of simulated data. For example, output of simulate_data
#'  @param effects_var A numeric vector specifying relative weights for a trajectory
#'  @param effect_size A number. the effect size to multiply effects_var by
#'  @param max_m A number.  The maximal numer of subjects to consider in calculations
#'  @param nsim A number. Number of simulations for simr
#'  @param alpha A number. The significance level for the power calculations
#'  @param tp A number. The time point of interest. The power calculation focuses on
#'  the ability to detect the effect in this time point
#'
#'  @return A power curve
get_simple_power_plot<-function(d,effects_vec,effect_size,
                                max_n=700,nsim=10,alpha=0.001,tp=1){
  simr_model = lmer(y~factor(t)+(1|g),data=d)
  # specify desired effect sizes
  for(j in 2:n_t){
    fixef(simr_model)[paste("factor(t)",j-1,sep="")] = effects_vec[j]*effect_size
  }
  # Analysis at a range of sample sizes
  model3 <- extend(simr_model, along="g", n=max_n)
  pc3 <- powerCurve(model3, along="g",
                    test=fixed(paste("factor(t)",tp,sep=""),"z"),
                    alpha=alpha,nsim=nsim)
  # plot(pc3,xlab="Number of subjects")
  return(pc3)
}

#' Power analysis that uses polynomials to model time trajectories.
#'
#'  @param d A data frame of simulated data. For example, output of simulate_data
#'  @param effects_var A numeric vector specifying relative weights for a trajectory
#'  @param effect_size A number. the effect size to multiply effects_var by
#'  @param max_m A number.  The maximal numer of subjects to consider in calculations
#'  @param nsim A number. Number of simulations for simr
#'  @param alpha A number. The significance level for the power calculations
#'  @param poly_deg A number. The polynomial degree. The power calculation focuses on
#'  the ability to detect the effect in this trajectory type
#'
#'  @return A power curve
get_poly_power_plot<-function(d,effects_vec,effect_size,
                              max_n=700,nsim=10,alpha=0.001,poly_deg=2){
 n_t = length(unique(d$t))
 pp = poly(0:(n_t-1),2)
 yy = effects_vec*effect_size
 new_effects_vec = lm(yy~pp)$coefficients[-1]
 rownames(pp) = as.character(0:(n_t-1))
 d_pp = pp[as.character(d$t),]
 rownames(d_pp) = NULL
 d = data.frame(d,d_pp)
 model = lmer(y~X1+X2+(1|g),data=d)
 fnames = summary(model)$coefficients
 fnames = rownames(fnames)[-1]
 simr_model = model
 # specify desired effect sizes
 for(j in 1:length(fnames)){fixef(simr_model)[fnames[j]] = new_effects_vec[j]}
 # Analysis at a range of sample sizes
 model3 <- extend(simr_model, along="g", n=max_n)
 pc3 <- powerCurve(model3, along="g",
                   test=fixed(fnames[poly_deg],"z"),
                   alpha=alpha,nsim=nsim)
 # plot(pc3,xlab="Number of subjects")
 return(pc3)
}

plot_ci_results<-function(l,cols = c("red","green","blue"),pchs=20:24,...){
  l = lapply(l,summary)
  plot(l[[1]][,1], l[[1]][,4],ylim=c(0,1.2),type="b",col=cols[1],pch=pchs[1],
       xlab="Number of subjects",ylab="power")
  for(j in 1:length(l)){
    x = l[[j]][,1]
    ui = pmin(1,l[[j]][,6])
    li = pmax(0,l[[j]][,5])
    arrows(x, li, x, ui, length=0.05, angle=90, code=3,col=cols[j])
    if(j>1){
      lines(l[[j]][,1], l[[j]][,4],type="b",col=cols[j],pch=pchs[j])
    }
  }
  abline(h=0.8,lty=2)
  legend(x="topleft",legend=names(l),pch=pchs,col=cols,lwd=2,ncol=2,...)
}
#' Auxiliary function for plotting power curves with errors.
#'
#'  @param l A list of powerCurve objects
#'  @param cols A vector of colors, length(cols)>=length(l)
#'  @param cols A vector of pch codes, length(pchs)>=length(l)
#'  @param ... Additional paramaters for legend
plot_ci_results<-function(l,cols = c("red","green","blue"),pchs=20:24,...){
  l = lapply(l,summary)
  plot(l[[1]][,1], l[[1]][,4],ylim=c(0,1.2),type="b",col=cols[1],pch=pchs[1],
       xlab="Number of subjects",ylab="power")
  for(j in 1:length(l)){
    x = l[[j]][,1]
    ui = pmin(1,l[[j]][,6])
    li = pmax(0,l[[j]][,5])
    arrows(x, li, x, ui, length=0.05, angle=90, code=3,col=cols[j])
    if(j>1){
      lines(l[[j]][,1], l[[j]][,4],type="b",col=cols[j],pch=pchs[j])
    }
  }
  abline(h=0.8,lty=2)
  legend(x="topleft",legend=names(l),pch=pchs,col=cols,lwd=2,ncol=3,...)
}
```

Results

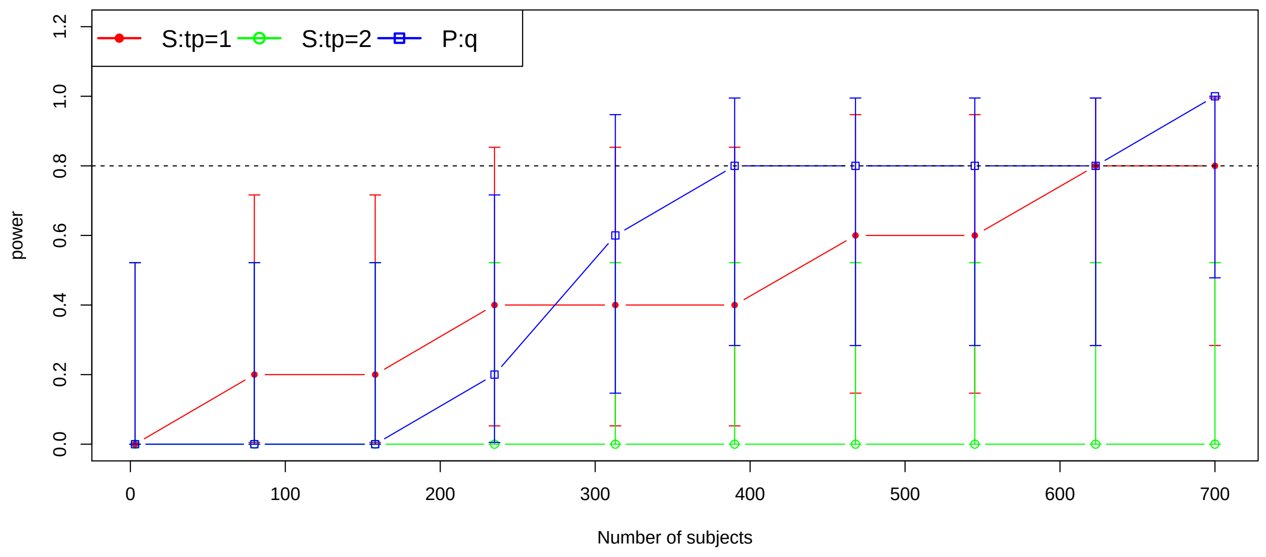

Figure 2: High within variance (sigma=1), peak effect size is 0.25

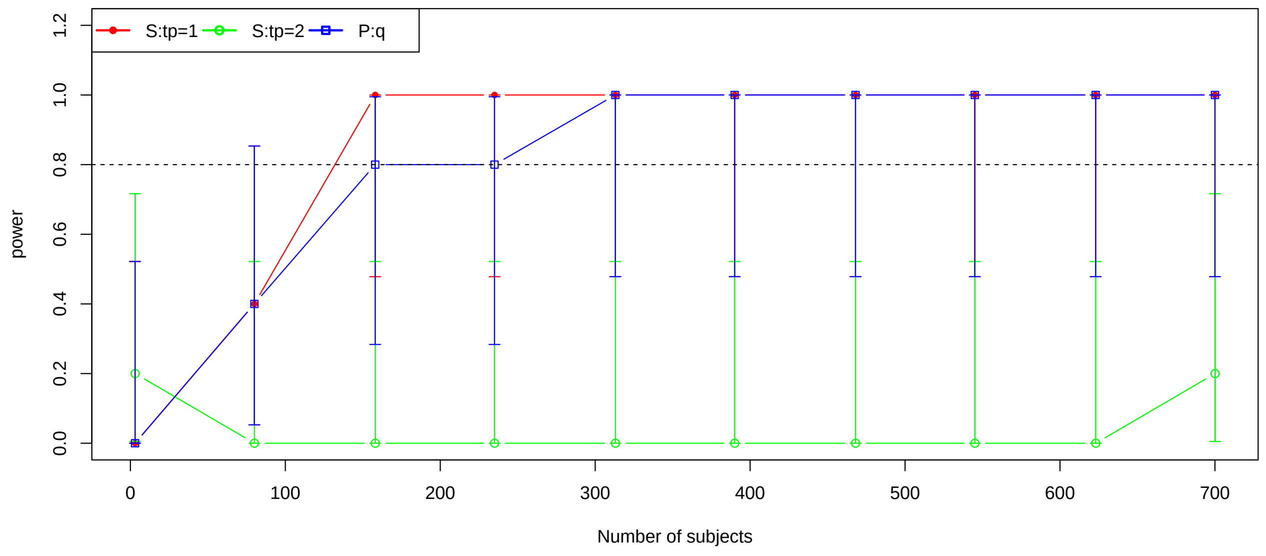

Figure 3: Medium within variance (sigma=0.5), peak effect size is 0.25