MoTrPAC power calculations

David Amar

9/15/2018

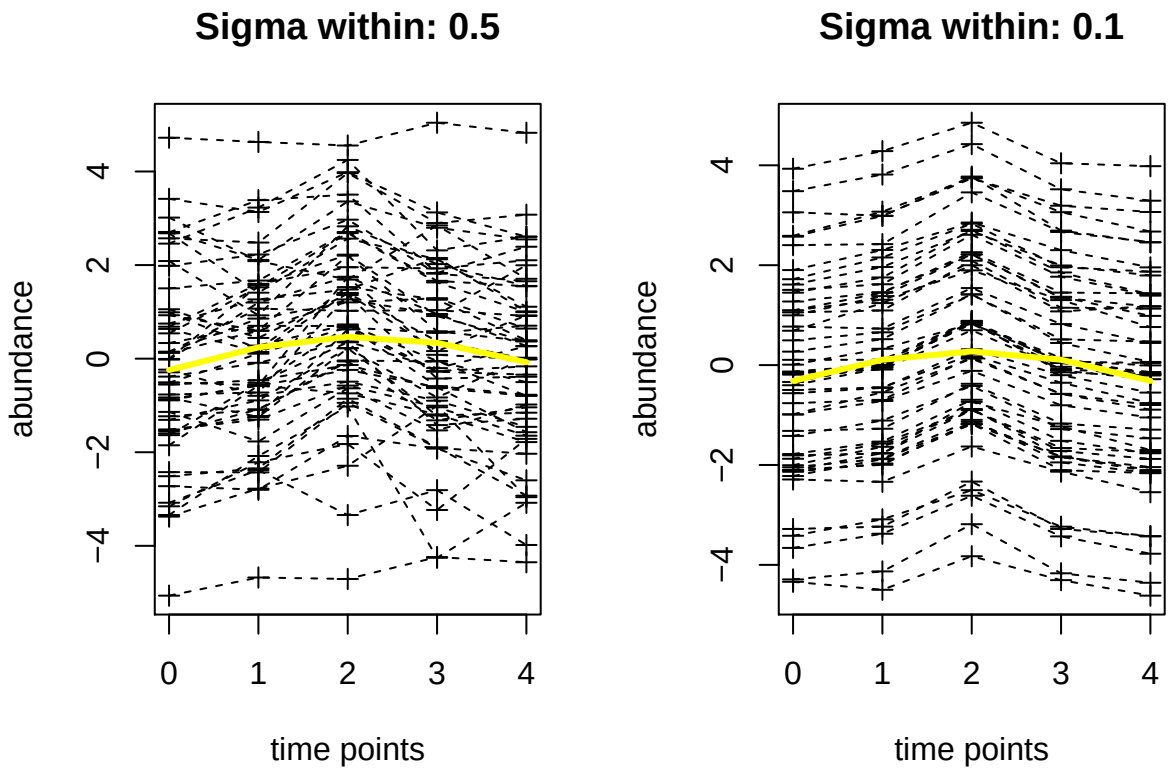
The underlying model

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 trend. We use the lme4 and simr packages to calculate the detection power for different parameters.

```
# 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),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
effects_vec = c(0,0.25,1,0.25,0)
effect_size = 1
par(mfrow=c(1,2))
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="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="Sigma within: 0.1")
```



Power calculations

We consider power calculations 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. The R code is given below.

```
library(lme4);library(simr)
```

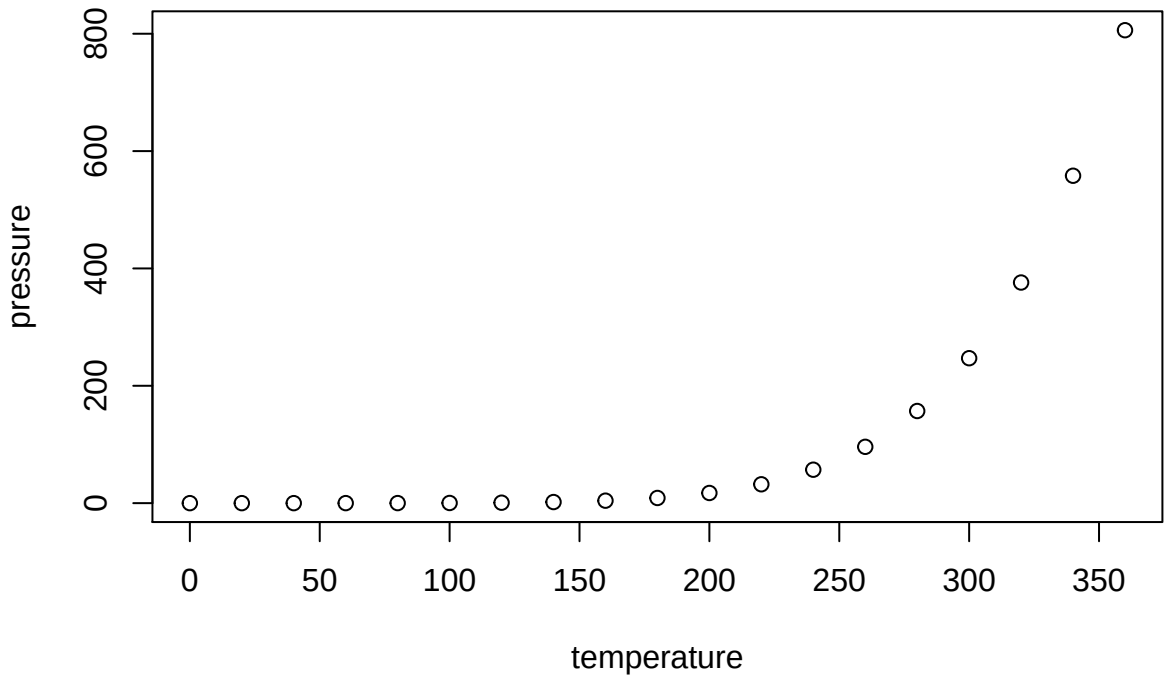
```
## Loading required package: Matrix
```

```
# Analysis that treats time as simple factors
get_simple_power_plot<-function(d,effects_vec,effect_size,max_n=700,nsim=10,alpha=0.001){
  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)",2,sep=""),"z"),
                    alpha=alpha,nsim=nsim)
  plot(pc3,xlab="Number of subjects")
  return(pc3)
}

get_poly_power_plot<-function(d,effects_vec,effect_size,max_n=700,nsim=10,alpha=0.001){
 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[2],"z"),
                   alpha=alpha,nsim=nsim)
 plot(pc3,xlab="Number of subjects")
 return(pc3)
}
```

1



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.