01 - Simulating data

2019-04-30

Here is some code to set up simulated data sets. The main aim of these simulations is to test the performance of phyloflows MCMC algorithm. Please read the sections “Our Job” and “Our Solution” on the main page before you go ahead here.

Data set 1 (simple, SARWS)

We start with simulating transmission flows between two population groups called “1” and “2”. We will assume a very simple sampling process, sampling at random within population strata, which we abbreviate to SARWS.

We first set up the sampling process within the two population groups. Let us assume population group 1 consists of 2000 individuals and group 2 of 2500 individuals, and that the sampling rates are 0.6 for group 1 and 0.45 for group 2. Here, we will suppose that sampling is at random within each of the population groups with these two sampling probabilities:

```
library(data.table)
set.seed(42)
ds <- data.table(CATEGORY=c(1,2),TRIAL=c(2000,2500),P_SEQ_EMP=c(0.6,0.45))
ds[,SUC:=TRIAL*P_SEQ_EMP]
ds
#>    CATEGORY TRIAL P_SEQ_EMP  SUC
#> 1:        1  2000      0.60 1200
#> 2:        2  2500      0.45 1125
```

Next, we calculate the sampling probabilities of transmission flows within and between the two groups.

```
dobs <- data.table(TR_TRM_CATEGORY=c(1,1,2,2),REC_TRM_CATEGORY=c(1,2,1,2))
tmp <- subset(ds,select=c('CATEGORY','P_SEQ_EMP'))
setnames(tmp,colnames(tmp),paste0('TR_TRM_',colnames(tmp)))
dobs <- merge(dobs,tmp,by='TR_TRM_CATEGORY')
setnames(tmp,colnames(tmp),gsub('TR_','REC_',colnames(tmp)))
dobs <- merge(dobs,tmp,by='REC_TRM_CATEGORY')
dobs[, S:= TR_TRM_P_SEQ_EMP * REC_TRM_P_SEQ_EMP]
set(dobs, NULL, c('TR_TRM_P_SEQ_EMP','REC_TRM_P_SEQ_EMP'), NULL)
setkey(dobs,TR_TRM_CATEGORY,REC_TRM_CATEGORY)
dobs
#>    REC_TRM_CATEGORY TR_TRM_CATEGORY      S
#> 1:                1               1 0.3600
#> 2:                2               1 0.2700
#> 3:                1               2 0.2700
#> 4:                2               2 0.2025
```

Next, we simulate true transmission flows. Let us assume 36% and 54% transmissions are within group 1 and 2 respectively, and 4% are from group 1 to group 2 and 6% are from group 2 to group 1:

$$\pi = (0.36, 0.04, 0.06, 0.54).$$

We further assume the total number of observed transmissions is $N = 300$. We will simulate the actual transmission count Z from a Poisson distribution. Then we will generate transmission flows between groups by

$$z \sim \text{Multinomial}(Z, \pi).$$

Finally we will generate observed transmissions flows by subsampling the actual transmission flows by

$$n_{ab} \sim \text{Binomial}(z_{ab}, \xi_{ab}), \forall a, b,$$

where ξ_{ab} is the probability of sampling a transmission event from a to b.

```r
TRUE_PI <- c(0.36,0.04,0.06,0.54)
N <- rpois(1,300/mean(dobs$S))
z <- rmultinom(1,size=N,prob=TRUE_PI)
n <- matrix(NA_integer_,ncol=1,nrow=length(TRUE_PI))
for (i in 1:length(TRUE_PI)){
    n[i] <- rbinom(1,size=z[i],dobs$S[i])
}
dobs[, TRM_OBS:= n]
dobs
#>    REC_TRM_CATEGORY TR_TRM_CATEGORY      S TRM_OBS
#> 1:                1               1 0.3600     139
#> 2:                2               1 0.2700      20
#> 3:                1               2 0.2700      15
#> 4:                2               2 0.2025     129
```

Corresponding phyloflows input data (simple, SARWS)

We will start by bringing the transmission count data into the form needed for phyloflows MCMC algorithm. We add an ID to each observation, called `TRM_CAT_PAIR_ID`. We also define the sampling groups. In this example, they correspond directly to the transmission groups.

```r
dobs[, TR_SAMPLING_CATEGORY:=TR_TRM_CATEGORY]
dobs[, REC_SAMPLING_CATEGORY:=REC_TRM_CATEGORY]
setkey(dobs,TR_TRM_CATEGORY,REC_TRM_CATEGORY)
dobs[, TRM_CAT_PAIR_ID:= seq_len(nrow(dobs))]
dobs[, S:=NULL]
dobs
#>    REC_TRM_CATEGORY TR_TRM_CATEGORY TRM_OBS TR_SAMPLING_CATEGORY REC_SAMPLING_CATEGORY TRM_CAT_PAIR_
#> 1:                1               1     139                    1                     1
#> 2:                2               1      20                    1                     2
#> 3:                1               2      15                    2                     1
#> 4:                2               2     129                    2                     2
```

We also need to define the prior distribution on the unknown sampling probabilities, and generate samples from it. At the very top of this page, defined the number of infected and sampled individuals in data.frame `ds`. Let us denote these by X^i_a and X^s_a for our two population groups a. Usually this type of information is available to us in real-world data analyses, and so we work from these numbers here also. Under the Binomial sampling model that we assume throughout, $X^s_a \sim Binom(X^i_a, \xi_a)$. If we suppose a flat prior on ξ_a, we obtain the posterior distribution of the sampling probabilities conditional on the number of total and sampled individuals,

$$p(\xi_a | X^i_a, X^i_s) = Beta(\xi_a; X^s_a + 1, X^i_a - X^s_a + 1).$$

This density typically contains a lot of information on the sampling process. To get this information into the form needed for **phyloflows** MCMC algorithm, we need to derive samples from that distribution. We also need to calculate their log density.

```
nprior<-1000
dprior<-ds[,list(P=rbeta(nprior,SUC+1,TRIAL-SUC+1),
         SAMPLE=1:nprior),
   by='CATEGORY']
dprior<-merge(dprior,ds,by='CATEGORY')
dprior[,LP:=dbeta(P,SUC+1,TRIAL-SUC+1,log=TRUE)]
set(dprior,NULL,c('TRIAL','SUC','P_SEQ_EMP'),NULL)
setnames(dprior, 'CATEGORY', 'SAMPLING_CATEGORY')
head(dprior)
#>    SAMPLING_CATEGORY         P SAMPLE       LP
#> 1:                 1 0.5824160      1 2.318750
#> 2:                 1 0.6184042      2 2.168504
#> 3:                 1 0.6033518      3 3.548540
#> 4:                 1 0.6015475      4 3.585452
#> 5:                 1 0.5918721      5 3.321375
#> 6:                 1 0.6034198      6 3.546614
```

This data set can be loaded through

```
data(twoGroupFlows1)
```

To test our MCMC algorithm, we repeated generating 100 such data set, and they can be loaded through

```
data(twoGroupFlows100)
```

Data set 2 (more complex, GLM)

We are now ready to consider a more complex example! We are still interested in transmission flows between the two groups "1" and "2". However in real-life sampling may depend on further characteristics of the populations in the two groups "1" and "2". If this is the case, the above SARWS assumption is obviously violated.

So where do we go from here? We could consider smaller sub-populations, though this runs the risk that estimates of the sampling probabilities quickly become very uncertain. And we know that if the sampling probabilities are uncertain, the performance of **phyloflows** MCMC engine will take a big hit. To avoid this problem, let us estimate the sampling probabilities for each sub-population based on a linear combination of predictive variables. We call this the GLM approach.

To keep things simple at the start, let us assume that sampling depends on gender within each group "1" and "2". We consider four population groups, "1M", "1F", "2M" and "2F". Let us assume there are 1,000 individuals in total, 40% individuals are in location 1, 60% individuals are in location 2, 60% individuals are women within each location, and that the sampling rates are 0.6 for men in group 1, 0.8 for women in group 1, 0.3 for men in group 2 and 0.7 for women in group 2:

```
library(data.table)
set.seed(42)
ds <- data.table(SEX=c(rep(c('M','F'),2)),
                 GROUP=c(rep(1,2),rep(2,2)),
                 P_SEQ=c(0.6,0.8,0.3,0.7)) # set up seq rates

ninf <- 1000
ninf1 <- round(ninf*0.4)
ninf2 <- ninf-ninf1
ninf1f <- round(ninf1*0.7)
ninf1m <- ninf1-ninf1f
ninf2f <- round(ninf2*0.6)
```

```
ninf2m <- ninf2-ninf2f
ds[,TRIAL:=c(ninf1m,ninf1f,ninf2m,ninf2f)]
ds[,SUC:=rbinom(nrow(ds),TRIAL,P_SEQ)]
ds[,CATEGORY:=paste0(GROUP,':',SEX)]
```

Next, we calculate the sampling probabilities of transmission flows within and between the two groups.

```
dobs <- data.table(TR_GROUP=c(rep(1,4),rep(2,4)),
                   REC_GROUP=c(rep(1,2),rep(2,2),rep(1,2),rep(2,2)),
                   TR_SEX=c(rep(c('F','M'),4)),
                   REC_SEX=c(rep(c('M','F'),4)))
dobs[,TR_TRM_CATEGORY:=paste0(TR_GROUP,":",TR_SEX)]
dobs[,REC_TRM_CATEGORY:=paste0(REC_GROUP,":",REC_SEX)]
dobs[, TRM_CAT_PAIR_ID:= seq_len(nrow(dobs))]
set(dobs, NULL, c('TR_GROUP','REC_GROUP','TR_SEX','REC_SEX'), NULL)
tmp <- subset(ds,select=c('CATEGORY','P_SEQ'))
setnames(tmp,colnames(tmp),paste0('TR_TRM_',colnames(tmp)))
dobs <- merge(dobs,tmp,by='TR_TRM_CATEGORY')
setnames(tmp,colnames(tmp),gsub('TR_','REC_',colnames(tmp)))
dobs <- merge(dobs,tmp,by='REC_TRM_CATEGORY')
dobs[, S:= TR_TRM_P_SEQ * REC_TRM_P_SEQ]
set(dobs, NULL, c('TR_TRM_P_SEQ','REC_TRM_P_SEQ'), NULL)
setkey(dobs,TRM_CAT_PAIR_ID)
dobs
#>    REC_TRM_CATEGORY TR_TRM_CATEGORY TRM_CAT_PAIR_ID    S
#> 1:              1:M             1:F               1 0.48
#> 2:              1:F             1:M               2 0.48
#> 3:              2:M             1:F               3 0.24
#> 4:              2:F             1:M               4 0.42
#> 5:              1:M             2:F               5 0.42
#> 6:              1:F             2:M               6 0.24
#> 7:              2:M             2:F               7 0.21
#> 8:              2:F             2:M               8 0.21
```

Next, we simulate true transmission flows between the sampling groups. Let us assume that 30% and 45% transmissions occur respectively within the groups "1" and "2". Further, 10% are from "1" to "2" and 15% are from "2" to "1". Of all those, 60% originate from men:

$$\pi = (0.12, 0.18, 0.04, 0.06, 0.06, 0.09, 0.18, 0.27).$$

We further assume the total number of observed transmissions is $N = 300$. We will simulate the actual transmission count Z from a Poisson distribution. Then we will generate transmission flows between groups by

$$z \sim \text{Multinomial}(Z, \pi).$$

Finally we will generate observed transmissions flows by subsampling the actual transmission flows by

$$n_{ab} \sim \text{Binomial}(z_{ab}, \xi_{ab}), \forall a, b,$$

where ξ_{ab} is the probability of sampling a transmission event from a to b.

```
TRUE_PI <- c(0.12,0.18,0.04,0.06,0.06,0.09,0.18,0.27)
N <- rpois(1,300/mean(dobs$S))
z <- rmultinom(1,size=N,prob=TRUE_PI)
n <- matrix(NA_integer_,ncol=1,nrow=length(TRUE_PI))
for (i in 1:length(TRUE_PI)){
    n[i] <- rbinom(1,size=z[i],dobs$S[i])
}
dobs[, TRM_OBS:= n]
dobs
#>    REC_TRM_CATEGORY TR_TRM_CATEGORY TRM_CAT_PAIR_ID    S TRM_OBS
#> 1:              1:M             1:F               1 0.48      55
#> 2:              1:F             1:M               2 0.48      77
#> 3:              2:M             1:F               3 0.24       7
#> 4:              2:F             1:M               4 0.42      22
#> 5:              1:M             2:F               5 0.42      17
#> 6:              1:F             2:M               6 0.24      28
#> 7:              2:M             2:F               7 0.21      35
#> 8:              2:F             2:M               8 0.21      56
```

Corresponding phyloflows input data (more complex, GLM)

We start again by bringing the transmission count data into the form needed for phyloflows MCMC algorithm. We add an ID to each observation, called `TRM_CAT_PAIR_ID`. We also define the sampling groups. As before, we set them to correspond to the transmission groups even if we are only interested in the flows within and between groups "1" and "2". After the model is fitted, we will aggregate the output to the flows between groups "1" and "2".

```
dobs[, TR_SAMPLING_CATEGORY:=TR_TRM_CATEGORY]
dobs[, REC_SAMPLING_CATEGORY:=REC_TRM_CATEGORY]
setkey(dobs,TR_TRM_CATEGORY,REC_TRM_CATEGORY)
dobs[, TRM_CAT_PAIR_ID:= seq_len(nrow(dobs))]
dobs[, S:=NULL]
dobs
#>    REC_TRM_CATEGORY TR_TRM_CATEGORY TRM_CAT_PAIR_ID TRM_OBS TR_SAMPLING_CATEGORY REC_SAMPLING_CATEGO
#> 1:              1:M             1:F               1      55                  1:F                  1
#> 2:              2:M             1:F               2       7                  1:F                  2
#> 3:              1:F             1:M               3      77                  1:M                  1
#> 4:              2:F             1:M               4      22                  1:M                  2
#> 5:              1:M             2:F               5      17                  2:F                  1
#> 6:              2:M             2:F               6      35                  2:F                  2
#> 7:              1:F             2:M               7      28                  2:M                  1
#> 8:              2:F             2:M               8      56                  2:M                  2
```

We also need to define the prior distribution on the unknown sampling probabilities, and generate samples from it. This final step is where the GLM approach differs from the SARWS approach. For the 4 groups

$$a = 1F, 1M, 2F, 2M$$

we model the sampling probability by

$$X_a^i \sim Binom(X_a^i, \xi_a)$$

using the same notation as above, and where

$$\operatorname{logit}(\xi_a) = \beta_0 + \beta_1 * G_a + \beta_2 S_a$$

where G_a equals 0 if the population group is "1", and 1 if the population group is "2", S_a equals 0 if gender is female and 1 if gender is male, and the priors on the regression coefficients are quite vague,

$$\beta_0 \sim N(0, 100), \beta_1 \sim N(0, 10), \beta_2 \sim N(0, 10).$$

Next we infer the posterior distribution of sampling probabilities with `stan`. The code for the above GLM regression model is as follows:

```
data{
    int<lower=1> N;
    int SUC[N];
    int TRIAL[N];
    int GROUP[N];
    int MALE[N];
}
parameters{
    real a;
    real grouptwo;
    real male;
}
model{
    vector[N] p_suc;
    male ~ normal( 0 , 10 );
    grouptwo ~ normal( 0 , 10 );
    a ~ normal( 0 , 100 );
    for ( i in 1:N ) {
        p_suc[i] = a + grouptwo * GROUP[i] + male * MALE[i];
    }
    SUC ~ binomial_logit( TRIAL , p_suc );
}
```

Let us fit the model to the sampling data, get samples from the posterior distributions, and calculate their log-density values:

```
library(rstan)
nprior <- 1e3
infile.sampling.stan.model <- 'glm_sex2comm2.stan' #in vignettes/glm_sex2comm2.stan
#   run STAN
tmp <- subset(ds,select=c('SEX','GROUP','TRIAL','SUC'))
tmp[, GROUP:=as.integer(GROUP==2)]
tmp[, MALE:= as.integer(SEX=='M')]
tmp <- as.list(tmp)
tmp$N <- nrow(ds)
fit.par <- stan(  file = infile.sampling.stan.model,
                  data = tmp,
                  iter = 2e3,
                  warmup = 5e2,
                  cores = 1,
                  chains = 1,
                  init = list(list(a=0, grouptwo=0, male=0)))
#>
#> SAMPLING FOR MODEL 'glm_sex2comm2' NOW (CHAIN 1).
#> Chain 1:
#> Chain 1: Gradient evaluation took 1.8e-05 seconds
#> Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
#> Chain 1: Adjust your expectations accordingly!
#> Chain 1:
#> Chain 1:
#> Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
#> Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
#> Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
#> Chain 1: Iteration:  501 / 2000 [ 25%]  (Sampling)
#> Chain 1: Iteration:  700 / 2000 [ 35%]  (Sampling)
#> Chain 1: Iteration:  900 / 2000 [ 45%]  (Sampling)
#> Chain 1: Iteration: 1100 / 2000 [ 55%]  (Sampling)
#> Chain 1: Iteration: 1300 / 2000 [ 65%]  (Sampling)
#> Chain 1: Iteration: 1500 / 2000 [ 75%]  (Sampling)
#> Chain 1: Iteration: 1700 / 2000 [ 85%]  (Sampling)
#> Chain 1: Iteration: 1900 / 2000 [ 95%]  (Sampling)
#> Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
#> Chain 1:
#> Chain 1:  Elapsed Time: 0.021588 seconds (Warm-up)
#> Chain 1:                0.064718 seconds (Sampling)
#> Chain 1:                0.086306 seconds (Total)
#> Chain 1:

# extract samples from the prior distribution
fit.e <- extract(fit.par)
tmp <- sample(length(fit.e$a), nprior)
ds[, GROUP:=as.integer(GROUP==2)]
ds[, MALE:= as.integer(SEX=='M')]
dprior <- ds[,
               {
                  z<- with(fit.e, a + grouptwo * GROUP + male * MALE)
                  list(SAMPLE=1:nprior, ETA=as.numeric(z[tmp]))
               },
               by=c('CATEGORY')]
dprior[, P:= exp(ETA)/(1+exp(ETA))]

# calculate log density using the betakernal bounded density estimator
require(bde)
tmp <- dprior[,
        {
            bdest<- bde( P,
                         dataPointsCache=sort(P),
                         b=0.001,
                         estimator='betakernel',
                         lower.limit=0,
                         upper.limit=1,
                         options = list( modified=FALSE,
                                         normalization='densitywise',
                                         mbc='none',
                                         c=0.5))
            list(SAMPLE=SAMPLE, LP=log(density(bdest, P)))
         },
         by='CATEGORY']
dprior <- merge(dprior, tmp, by=c('CATEGORY','SAMPLE'))
set(dprior, NULL, c('ETA'), NULL)
setnames(dprior, 'CATEGORY', 'SAMPLING_CATEGORY')
head(dprior)
#>    SAMPLING_CATEGORY SAMPLE         P       LP
#> 1:               1:F      1 0.8078519 2.651191
#> 2:               1:F      2 0.8306516 2.780450
#> 3:               1:F      3 0.8156518 2.785355
#> 4:               1:F      4 0.8289016 2.798152
#> 5:               1:F      5 0.8402540 2.583188
#> 6:               1:F      6 0.8323180 2.758882
```

This data set can be loaded through

```
data(fourGroupFlows1)
```