phyloflows: Practical example

Xiaoyue Xi and Oliver Ratmann

2021-06-30

This vignette describes the analysis on real-world HIV data from Uganda. The analysis was explained in `Inferring the sources of HIV infection in Africa from deepsequence data with semi-parametric Bayesian Poisson flow models` (link to appear).

Getting started

1. **RCCS_metadata** includes individual-level meta-data on sequenced individuals. This adds ages and locations to the data.

```
library(data.table)
df <- read.csv('~/phyloscanner/phyloflows/data/RCCS_metadata.csv')
df <- data.table(df)
df[,X:=NULL]
head(df)
#>           ID SEX AGE_AT_MID COMM_TYPE INMIG_2YRS_LOC
#> 1: RkA00001F   F     40.731         i       resident
#> 2: RkA00008F   F     48.805         i       resident
#> 3: RkA00116M   M     49.990         i       resident
#> 4: RkA00118M   M     41.417         i       resident
#> 5: RkA00122F   F     42.217         i       resident
#> 6: RkA00124F   F     40.111         i       resident
```

- *ID*: individual IDs
- *SEX*: genders
- *AGE_AT_MID*: ages at the mid of study
- *COMM_TYPE*: locations (f - fishing sites, i - inland)
- *INMIG_2YRS_LOC*: immigration status in 2 years

2. **RCCS_pairs** includes the transmission pairs reconstructed from Phyloscanner.R.utilities.

```
rtr <- read.csv('~/phyloscanner/phyloflows/data/RCCS_pairs.csv')
rtr <- data.table(rtr)
rtr[,X:=NULL]
head(rtr)
#>        TR_ID    REC_ID
#> 1: RkA06116M RkA00124F
#> 2: RkA02891M RkA00158F
#> 3: RkA07616M RkA00204F
#> 4: RkA06489M RkA00263F
#> 5: RkA06557F RkA00282M
#> 6: RkA02324M RkA00324F
```

- *TR_RID*: source IDs
- *REC_RID*: recipient IDs

Pre-processing

Meta data and reconstructed pairs

We process the pair data and meta data, in order to group individuals into sub-populations and count transmissions between sub-populations. In particular, the subpopulation is defined by one-year age bands, genders and locations. Also, we considered the locations of sources to be the origin of inmigrantion, if the reconstructed sources migrated within 2 years.

```r
library(data.table)
setnames(df,colnames(df),paste0('TR_',colnames(df)))
rtr <- merge(rtr, df, by='TR_ID')
setnames(df,colnames(df),gsub('TR_','REC_',colnames(df)))
rtr <- merge(rtr, df, by='REC_ID')

# set unknown source location to fishing
rtr[is.na(TR_COMM_TYPE),TR_COMM_TYPE:='f']
# set unknown recipient location to inland
rtr[is.na(REC_COMM_TYPE),REC_COMM_TYPE:='i']



# impute age
tmp <- which(is.na(rtr$TR_AGE_AT_MID))
set(rtr, tmp, 'TR_AGE_AT_MID', mean(rtr$TR_AGE_AT_MID[which(!is.na(rtr$TR_AGE_AT_MID))]) )
tmp <- which(is.na(rtr$REC_AGE_AT_MID))
set(rtr, tmp, 'REC_AGE_AT_MID', mean(rtr$REC_AGE_AT_MID[which(!is.na(rtr$REC_AGE_AT_MID))]) )

# set unknown migration to residents
rtr[is.na(TR_INMIG_2YRS_LOC), TR_INMIG_2YRS_LOC:='resident']
rtr[TR_INMIG_2YRS_LOC=='resident',TR_COMM_TYPE_MIG:=TR_COMM_TYPE]
tmp <- rtr[, which(TR_INMIG_2YRS_LOC=='unknown')]

#   set unknown origin to fishing
set(rtr, tmp, 'TR_COMM_TYPE_MIG', 'f')

# set known migration
rtr[TR_INMIG_2YRS_LOC=='inland',TR_COMM_TYPE_MIG:='i']
rtr[TR_INMIG_2YRS_LOC=='fish',TR_COMM_TYPE_MIG:='f']
rtr[TR_INMIG_2YRS_LOC=='external',TR_COMM_TYPE_MIG:='e']

#   stratify age
rtr[, TR_AGE_AT_MID_C:= as.character(cut(TR_AGE_AT_MID, breaks=c(15,16:49,50), labels=paste0(15:49,'-',
rtr[, REC_AGE_AT_MID_C:= as.character(cut(REC_AGE_AT_MID, breaks=c(15,16:49,50), labels=paste0(15:49,'-
rtr <- subset(rtr, !is.na(REC_AGE_AT_MID_C))

#   build category to match with sampling data tables
rtr[, REC_SAMPLING_CATEGORY:= paste0(REC_COMM_TYPE,':',REC_SEX,':',REC_AGE_AT_MID_C)]
rtr[, TR_SAMPLING_CATEGORY:= paste0(TR_COMM_TYPE,':',TR_SEX,':',TR_AGE_AT_MID_C)]

#   build transmission flow category
rtr[, REC_TRM_CATEGORY:= paste0(REC_COMM_TYPE,':',REC_SEX,':',REC_AGE_AT_MID_C)]
rtr[, TR_TRM_CATEGORY:= paste0(TR_COMM_TYPE_MIG,':',TR_SEX,':',TR_AGE_AT_MID_C)]

# make all combinations of variables
dac <- expand.grid(COMM_TYPE_F= c('i','f'), SEX=  c('F','M'), AGE_AT_MID_C= paste0(15:49,'-',16:50))
dac <- as.data.table(dac)
dac[, CATEGORY:= paste0(COMM_TYPE_F, ':', SEX, ':', AGE_AT_MID_C)]
dac <- as.data.table(expand.grid(TR_SAMPLING_CATEGORY= dac$CATEGORY, REC_SAMPLING_CATEGORY= dac$CATEGORY

# ignore Male-Male and Female-Female combinations
dac <- subset(dac, !(grepl('F',TR_SAMPLING_CATEGORY)&grepl('F',REC_SAMPLING_CATEGORY)) & !(grepl('M',TR_

# add transmission categories
dac[, REC_TRM_CATEGORY:= REC_SAMPLING_CATEGORY]
dac[, TR_TRM_CATEGORY:= TR_SAMPLING_CATEGORY]

# add inmigrants from external communities
tmp <- copy(dac)
set(tmp, NULL, 'TR_TRM_CATEGORY', tmp[,gsub('^[f|i]','e',TR_SAMPLING_CATEGORY)])
dac <- rbind(dac, tmp)

# add inmigrants sampled in inland and migrated from fishing
tmp <- dac[grepl('^i',TR_SAMPLING_CATEGORY)]
set(tmp, NULL, 'TR_TRM_CATEGORY', tmp[,gsub('^i','f',TR_SAMPLING_CATEGORY)])
dac <- rbind(dac, tmp)

# add inmigrants sampled in fishing and migrated from inland
tmp <- dac[grepl('^f',TR_SAMPLING_CATEGORY)]
set(tmp, NULL, 'TR_TRM_CATEGORY', tmp[,gsub('^f','i',TR_SAMPLING_CATEGORY)])
dac <- rbind(dac, tmp)

# remove duplicated rows
# TR_SAMPLING_CATEGORY f: F: 15-24: 1 and i: F: 15-24: 1 are all set to e: F: 15-24: 1
dac <- unique(dac)

#   calculate observed number of transmissions
dobs    <- rtr[, list( TRM_OBS=length(unique(TR_ID))), by=c('TR_TRM_CATEGORY','REC_TRM_CATEGORY','TR_SAM
dac[, DUMMY:= 1]
dobs <- merge(dac, dobs, by=c('TR_TRM_CATEGORY', 'REC_TRM_CATEGORY','TR_SAMPLING_CATEGORY', 'REC_SAMPLI
stopifnot( dobs[, !any(is.na(DUMMY))] )
set(dobs, NULL, 'DUMMY', NULL)
set(dobs, dobs[, which(is.na(TRM_OBS))], 'TRM_OBS', 0L)

#   make SMOOTH_CATEGORY
dobs[,TR_SMOOTH_CATEGORY:= as.numeric(substr(TR_TRM_CATEGORY,5,6))+0.5]
dobs[,REC_SMOOTH_CATEGORY:= as.numeric(substr(REC_TRM_CATEGORY,5,6))+0.5]
dobs[,OUTPUT:=paste0(substr(TR_TRM_CATEGORY,1,1),':',
                     substr(TR_TRM_CATEGORY,3,3),':',
                     substr(TR_SAMPLING_CATEGORY,1,1),':',
                     substr(REC_TRM_CATEGORY,1,1),':',
                     substr(REC_TRM_CATEGORY,3,3))]
tmp <- subset(dobs,select = 'OUTPUT')
tmp <- unique(tmp)
setkey(tmp, OUTPUT)
tmp[, OUTPUT_ID:= seq_len(nrow(tmp))]
dobs <- merge(dobs, tmp, by='OUTPUT')
setkey(dobs,OUTPUT_ID,TR_SMOOTH_CATEGORY,REC_SMOOTH_CATEGORY)
```

```
# make PAIR_ID
dobs[, TRM_CAT_PAIR_ID:= seq_len(nrow(dobs))]
setkey(dobs, TRM_CAT_PAIR_ID)
```

Sampling data

We load samples drawn from the distribution of sampling fractions, as described in the basic example. Additional standard models to **phyloflows** were used to produce the samples, and details can be found in the paper. You may build other types of models for sampling.

```
dprior <- read.csv('~/phyloscanner/phyloflows/data/RCCS_sampling.csv')
dprior <- data.table(dprior)
dprior[,X:=NULL]
head(dprior)
```

Flow inference

Pre-processing gives the data needed for flow inference, including **dobs** and **dprior**. Now we are ready to use **phyloflows** to estimate transmission flows between subpopulations.

Independent models

The first model considers flows to be independent between sub-populations, and estimates flows from the basic command in **phyloflow** through MCMC.

```
library(phyloflows)
mcmc.file   <- 'mcmc'
control <- list(seed=42,
                mcmc.n=nrow(unique(dprior[,c('SAMPLING_CATEGORY','WHO')])) * 2 * 1e5,
                verbose=0,
                outfile=mcmc.file,
                sweep_group=1e4L)
source.attribution.mcmc(dobs, dprior, control=control)
```

Correlated flows by ages

The second model treats flows correlated among the neighbouring age groups. We estimate flows by `Stan` through HMC. Note than `Stan` does not support samples drawn from distributions as inputs. Instead, we fit Beta distributions to samples in **dprior**. This model was facilated by reduced-rank Gaussian process approximation implemented in the file.

```
# fit beta to dprior
library(MASS)
dprior.fit <- dprior[,{tmp <- fitdistr(P, "beta", start=list(shape1=1,shape2=1/mean(P)),lower=c(0,0))
  list(SHAPE1=tmp$estimate[1],SHAPE2=tmp$estimate[2])
  },by=c('SAMPLING_CATEGORY','WHO')]

# index sampling groups
dprior.id <- subset(dprior.fit,select = c('SAMPLING_CATEGORY','WHO'))
setkey(dprior.id, SAMPLING_CATEGORY,WHO)
dprior.id[,ID:= seq_len(nrow(dprior.id))]
tmp <- subset(dobs, select = c('TR_SAMPLING_CATEGORY',
                               'REC_SAMPLING_CATEGORY',
                               'TRM_CAT_PAIR_ID'))
setnames(dprior.id,colnames(dprior.id),paste0('TR_',colnames(dprior.id)))
```

```
tmp <- merge(tmp,subset(dprior.id[TR_WHO=='TR_SAMPLING_CATEGORY',],select=c('TR_ID','TR_SAMPLING_CATEGO
setnames(dprior.id,colnames(dprior.id),gsub('TR_','REC_',colnames(dprior.id)))
tmp <- merge(tmp,subset(dprior.id[REC_WHO=='REC_SAMPLING_CATEGORY',],select=c('REC_ID','REC_SAMPLING_CA
setkey(tmp, TRM_CAT_PAIR_ID)
xi_id <- cbind(tmp$TR_ID, tmp$REC_ID)


# GP approximation
M <- 30
D <- 2
indices <- matrix(NA, M^D, D)
mm=0;
for (m1 in 1:M){
  for (m2 in 1:M){
    mm = mm+1
    indices[mm,] = c(m1, m2)
  }
}

L <-  matrix(rep(c(49.5, 49.5) * 5/4,each=nrow(indices)),nrow=nrow(indices))
sevalue <- pi * indices / (2 * L)
efunc <-  do.call( rbind, lapply(1:1225, function(k){as.vector(apply(sqrt(1/L) * sin(sevalue *(matrix(r


# stan input
data.fit <- list(N=nrow(dobs),
                N_group = max(dobs$OUTPUT_ID),
                N_per_group = 1225,
                y=matrix(dobs$TRM_OBS,nrow=1225,ncol=max(dobs$OUTPUT_ID)),
                D=2,
                x=cbind(dobs$TR_SMOOTH_CATEGORY[1:1225],dobs$REC_SMOOTH_CATEGORY[1:1225]),
                M_nD=900,
                Xgp=efunc,
                slambda=sevalue,
                N_xi = nrow(dprior.fit),
                shape = cbind(dprior.fit$SHAPE1,dprior.fit$SHAPE2),
                xi_id_src = matrix(xi_id[,1],nrow=1225,ncol=max(dobs$OUTPUT_ID)),
                xi_id_rec = matrix(xi_id[,2],nrow=1225,ncol=max(dobs$OUTPUT_ID)),
                id_mf =   dobs[grepl(':M:',TR_TRM_CATEGORY),unique(OUTPUT_ID)],
                id_fm =   dobs[grepl(':F:',TR_TRM_CATEGORY),unique(OUTPUT_ID)],
                id_eh = dobs[grepl('e:',TR_TRM_CATEGORY) & grepl('f:',REC_TRM_CATEGORY),unique(OUTPUT_I
                id_el = dobs[grepl('e:',TR_TRM_CATEGORY) & grepl('i:',REC_TRM_CATEGORY),unique(OUTPUT_I
                id_hh = dobs[grepl('f:',TR_TRM_CATEGORY) & grepl('f:',REC_TRM_CATEGORY),unique(OUTPUT_I
                id_hl = dobs[grepl('f:',TR_TRM_CATEGORY) & grepl('i:',REC_TRM_CATEGORY),unique(OUTPUT_I
                id_lh = dobs[grepl('i:',TR_TRM_CATEGORY) & grepl('f:',REC_TRM_CATEGORY),unique(OUTPUT_I
                id_mf_h =  dobs[grepl(':M:',TR_TRM_CATEGORY) & grepl('f:',REC_TRM_CATEGORY),unique(OUTP
                id_mf_l =  dobs[grepl(':M:',TR_TRM_CATEGORY) & grepl('i:',REC_TRM_CATEGORY),unique(OUTP
                id_fm_h =  dobs[grepl(':F:',TR_TRM_CATEGORY) & grepl('f:',REC_TRM_CATEGORY),unique(OUTP
                id_fm_l =  dobs[grepl(':F:',TR_TRM_CATEGORY) & grepl('i:',REC_TRM_CATEGORY),unique(OUTP
                id_sh =  dobs[grepl('f:',TR_TRM_CATEGORY),unique(OUTPUT_ID)],
                id_sl =  dobs[grepl('i:',TR_TRM_CATEGORY),unique(OUTPUT_ID)],
                id_se =  dobs[grepl('e:',TR_TRM_CATEGORY),unique(OUTPUT_ID)])
```

```r
# run stan
library(rstan)
fit <- stan(file = file.path('gpa4.stan'),
            data = data.fit,
            iter = 3000,  warmup = 500, chains=1, thin=1, seed = 42,
            algorithm = "NUTS", verbose = FALSE,
            control = list(adapt_delta = 0.999,max_treedepth=15))
```

Going forward

So far, we have covered how to reproduced the main analysis in the paper. Next, please use your R wizadry or other functions in **phyloflows** to process the output further and answer practical questions in disease surveillance.