# Estimating transmission flows under heterogeneous sampling - a first example

*2019-04-30*

This vignette gives a basic first introduction to estimating transmission flows with the `phyloflows` package.

## Input data

**phyloflows** expects input data in a specific format.

- `dobs` a data.frame of observed transmission counts within and between population groups.
- `dprior` a data.frame that summarises prior information on how population groups were sampled.

To get you started, **phyloflows** comes with a small simulated example data set of transmission counts and sampling information between two population groups, denoted by "1" and "2":

```
#   required R packages
require(phyloflows)
require(ggplot2)
require(bayesplot)
require(data.table)
require(coda)

#
#   load transmission flow data "twoGroupFlows1"
data(twoGroupFlows1, package="phyloflows")
#   observed transmission counts
dobs <- twoGroupFlows1$dobs
#   sampling information
dprior <- twoGroupFlows1$dprior
```

### Input data: observed transmission flows

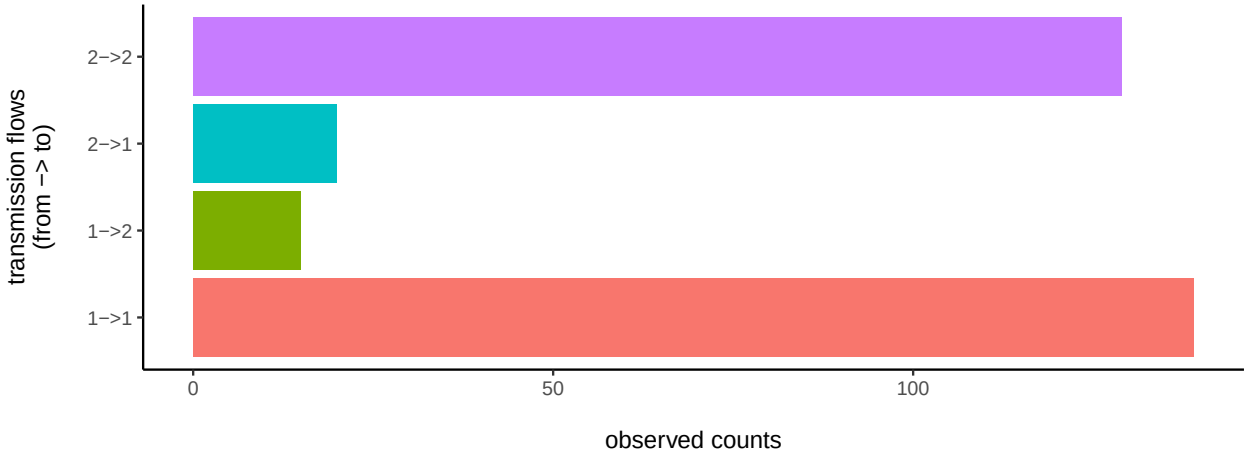
```
dobs
#>    REC_TRM_CATEGORY TR_TRM_CATEGORY TR_SAMPLING_CATEGORY
#> 1:                1               1                     1
#> 2:                2               1                     1
#> 3:                1               2                     2
#> 4:                2               2                     2
#>    REC_SAMPLING_CATEGORY TRM_OBS TRM_CAT_PAIR_ID
#> 1:                     1     139                1
#> 2:                     2      15                2
#> 3:                     1      20                3
#> 4:                     2     129                4
```

**`dobs` specifies observed counts of transmissions from a transmitter group to a recipient group.** It must contain the following columns:

- *TR_TRM_CATEGORY* name of transmitter group.
- *REC_TRM_CATEGORY* name of recipient group.

- *TRM_CAT_PAIR_ID* identifier of transmitter-recipient pair
- *TRM_OBS* observed transmission counts

Let us look at the data. The first row contains counts of transmission flows from group "1" to group "1", and there are 139 of them. The next row contains counts of transmission flows from group "1" to group "2", and there are 15 of them. Here is a barplot of our input data:




## Input data: sampling information

**`dobs` also must contain information about how each group was sampled.** This is stored in the following columns:

- *TR_SAMPLING_CATEGORY* sampling strata of transmitter group
- *REC_SAMPLING_CATEGORY* sampling strata of recipient group

Each transmitter/recipient group is associated to a sampling category. This can be "sampling group a" for both "1" and "2", or "a" and "b" respectively for "1" and "2". In our little data set, we gave the same name to transmitter/recipient and sampling groups.

**`dprior` specifies the probability of sampling an individual from each sampling group.** To keep this as flexible as possible, samples from the sampling distribution, rather than say the mean and standard deviation, need to be given. This information is stored in the following columns:

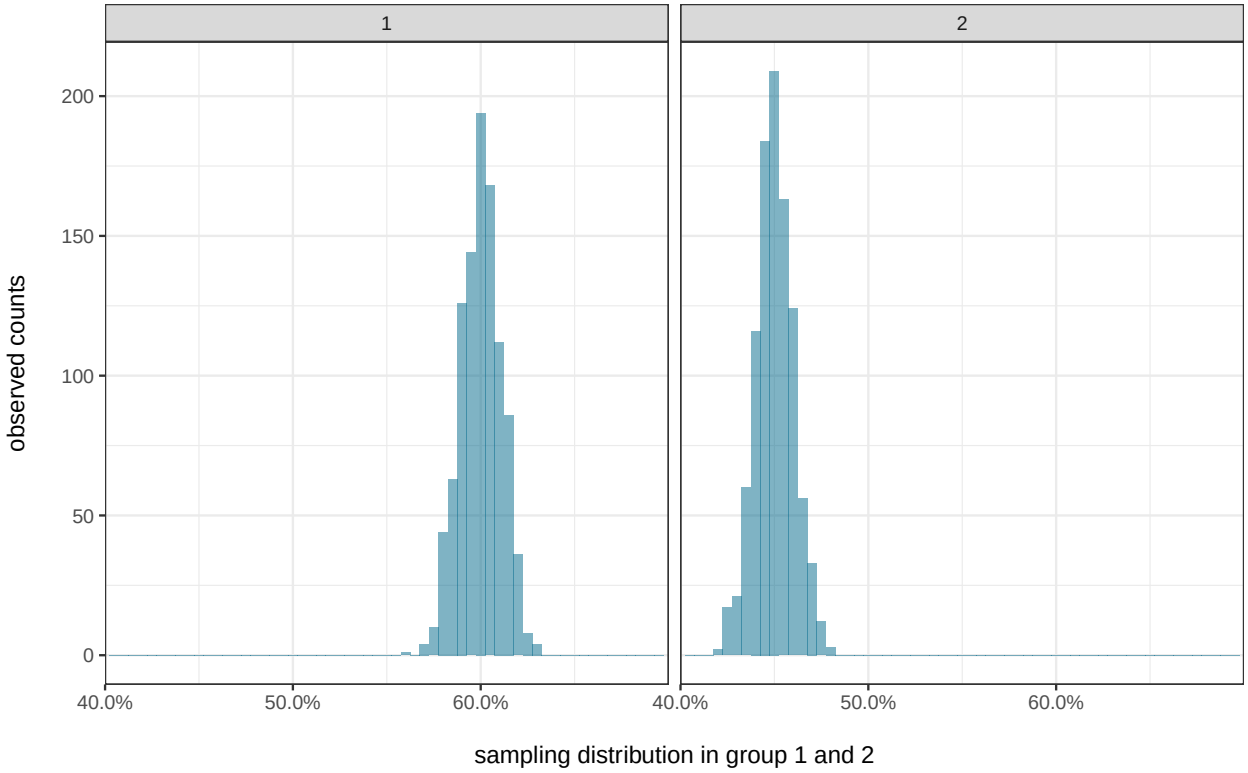
- *SAMPLING_CATEGORY* name of sampling strata
- *SAMPLE* identifier of sample from the sampling distribution
- *P* sampling probability
- *LP* log density of the sampling probability under the sampling distribution.

Let us look at the sampling information:

```
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
```

Here is a histogram of the sampling distribution from sampling groups "1" and "2". Notice that in our example, the probability of sampling individuals in group "1" is higher than that among individuals in group

"2".




## Statistical model

`phyloflows` **uses a Bayesian approach to estimate the proportion of transmissions** between the two population groups,

$$\pi = (\pi_{11}, \pi_{12}, \pi_{21}, \pi_{22}).$$

The model can be motivated as follows. Suppose the actual, unobserved number of transmissions from group $i$ to group $j$ are $z_{ij}$. Denote the vector of actual transmission counts by

$$z = (z_{11}, z_{12}, z_{21}, z_{22}).$$

We assume that transmission events occurred independently of each other. Then the likelihood of the actual transmission counts can be modelled by

$$p(z|Z, \pi) = Multinomial(z; Z, \pi),$$

where $Z$ is the total number of transmissions, $Z = \sum_{kl} z_{kl}$. Next, we specify a model for observing one actual transmission event. We assume that sampling occurred at random within each of the sampling groups $i$ and $j$. We then obtain

$$p(n_{ij}|z_{ij}, s_i, s_j) = Binomial(n_{ij}; z_{ij}, s_i * s_j),$$

where $s_i$ is the probability of sampling an individual from group $i$, and similary for $s_i$.

These equations suggest that one approach to infer the proportion of transmissions $\pi$ could be via data augmentation. In data augmentation, we would consider the unobserved, actual transmission counts $z$ as latent variables, and then infer the joint posterior distribution of the parameters $(z, Z, \pi)$ with a Monte Carlo algorithm.

However there is a more efficient approach for the particular model above. Inference of $\pi$ under the Multinomial likelihood $Multinomial(z; Z, \pi)$ is equivalent to inference of Poisson mean rates $\lambda$

$$\lambda = (\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22})$$

in the system of independent Poisson likelihoods

$$p(z_{ij}|\lambda_{ij}) = Poisson(z; \lambda_{ij}),$$

where $\lambda_{ij} > 0$, $i = 1, 2$ and $j = 1, 2$. The proportion of transmissions $\pi$ are recovered via the equations

$$\pi_{ij} = \lambda_{ij} / \sum_{k=1,2;l=1,2} \lambda_{kl}.$$

for $i = 1, 2$ and $j = 1, 2$. This is known as the Poisson trick. The advantage of this model parameterisation is that sampled Poisson random variables are again Poisson random variables, which allows us to integrate out analytically the unknown, actual transmission counts $z_{ij}$. We obtain

$$p(n|\lambda, s) = \prod_{i=1,2;j=1,2} Poisson(n_{ij}; \lambda_{ij} * s_i * s_j).$$

The free parameters of the model are $(\lambda, s)$, and the posterior distribution of the free parameters is given by

$$\begin{aligned} p(\lambda, s|n) &\propto p(n|\lambda, s)p(\lambda, s) \\ &= \prod_{i=1,2;j=1,2} Poisson(n_{ij}; \lambda_{ij} * s_i * s_j)p(\lambda_{ij})p(s_i)p(s_j). \end{aligned}$$

For the prior distributions, we specify:

- $p(\lambda_{ij})$: an uninformative prior distribution. We use a Gamma distribution with parameters $\alpha_i = 0.8/4$ and $\beta = 0.8/Z$ with $Z = \sum_{ij|n_{ij}>0} n_{ij}/(s_i * s_j) + \sum_{ij|n_{ij}>0}(1 - s_i * s_j)/(s_i * s_j)$. This choice implies for $\pi$ a Dirichlet prior distribution with parameters $\alpha_i$, which is considered to be an objective choice.
- $p(s_i)$: a strongly informative prior distribution, based on available data (as illustrated above).

# MCMC

## MCMC: syntax

We use a Markov Chain Monte Carlo algorithm to sample from the posterior distribution

$$p(\lambda, s|n) \propto \prod_{i=1,2;j=1,2} Poisson(n_{ij}; \lambda_{ij} * s_i * s_j)p(\lambda_{ij})p(s_i)p(s_j).$$

Then, we calculate the main quantity of interest, $\pi$, via

$$\pi_{ij} = \lambda_{ij} / \sum_{k=1,2;l=1,2} \lambda_{kl}.$$

for $i = 1, 2$ and $j = 1, 2$. The syntax for running the algorithm is as follows.

```
# specify a list of control variables:
#   seed    random number seed
#   mcmc.n  number of MCMC iterations
#   verbose flag for verbose output
#   outfile output file name if you like to have the results
#           written to an *.rda* file
control <- list(seed=42, mcmc.n=500, verbose=0)
# run MCMC
ans <- source.attribution.mcmc(dobs, dprior, control)
```

## MCMC: messages

Let s have a look at the messages first.

- The total number of unknown parameters in **phyloflows** MCMC is the length of $\pi$ plus the length of the latent transmission flows $z$ plus the length of the pairwise sampling probabilities $\xi$, $\xi_{ij} = s_i * s_j$, plus twice the length of the sampling probabilities $s$, plus 1 for $Z$. This makes $4 + 4 + 4 + 2 * 2 + 1 = 17$ in our very first example.
- The MCMC updates all these parameters in a certain number of MCMC iterations, and this number is called a sweep. A sweep is always twice the length of the sampling probabilities plus 1 for updating the values of $\pi$. This makes $2 * 2 + 1$ in our very first example.
- The total number of sweeps is determined from `control[['mcmc.n']]`. In our case, it is $50/5 = 10$. If we had set `control[['mcmc.n']]<- 51`, then the total number of sweeps would have been 11.
- The total number of iterations is the length of a sweep times the total number of sweeps. In our example, $5 * 10 = 50$. If we had set `control[['mcmc.n']]<- 51`, then the total number of iterations would have been $5 * 11 = 55$.
- Finally, we have the number of transmission pair categories updated per iteration. These numbers are important to assess the likely performance of the MCMC. The algorithm proceeds by updating the sampling probabilities for the transmitter groups, then those for the recipient groups, and finally one update for the values of $\pi$. If we update $s_1$ for the transmitter groups, we also need to update all pairwise sampling probabilities that contain $s_1$ for the transmitter groups. These are $\xi_{11} = s_1 * s_1$ and $\xi_{12} = s_1 * s_2$, so we have two pairwise sampling probabilities to update. In our very first example, this is always the case: if we update any of the 4 $s_i$ for either the transmitter and recipient groups, we always have to update 2 pairwise sampling probabilities. This is what you see printed, before the algorithm gets cranking. In general, the fewer pairwise sampling probabilities need to be updated, the better, because the MCMC acceptance rates take a huge hit when many parameters need to be updated at once. Set up your model so that you have at most 4-6 joint parameter updates at any MCMC iteration.

## MCMC: output

Let us have a look at the output:

```
str(ans)
```

We are mostly interested in the joint posterior distribution

$$p(\pi|n),$$

which is just a component of the entire posterior distribution of all parameters

$$p(\pi, z, Z, \xi, s|n).$$

So let us look at just this component in the output, and make a trace plot:

```
post.pi <- ans[['pars']][['PI']]
colnames(post.pi) <- paste0('PI-',1:ncol(post.pi))
bayesplot:::mcmc_trace(post.pi, pars=colnames(post.pi), facet_args = list(ncol = 1), n_warmup=0)
```

Ok, fab. Of course we would like many more iterations, perhaps $10,000$ sweeps are a good number. We can do that. But what really are *PI-1*, *PI-2*, *PI-3*, *PI-4*? The numbers 1-4 are just the values of the transmission pair IDs in `dobs`, `dobs$TRM_CAT_PAIR_ID`. So we can associate more interpretable names to the output as follows:

```
post.pi <- ans[['pars']][['PI']]
setkey(dobs, TRM_CAT_PAIR_ID) #order by pair IDs
post.pi.colnames <- paste0('PI ',dobs$TR_TRM_CATEGORY,'->',dobs$REC_TRM_CATEGORY)
```

```
colnames(post.pi) <- post.pi.colnames
bayesplot:::mcmc_trace(post.pi, pars=colnames(post.pi), facet_args = list(ncol = 1), n_warmup=0)
```

That's it for now. Use your usual R wizadry to process the output further, and have a look at the other vignettes.