Analyses for Estimating Abundance From Genetic Mark-Recapture Data When Not All Sites Are Sampled

Tim Frasier

April 5, 2019

1 Computer Set-Up

To run these analyses correctly, a number of R packages and other programs must be installed and working on your computer. These are:

1. R - link
2. RStudio - link
3. R Packages:
   (a) rstan
   (b) ggplot2
4. Stan - link

All other needed code is located within the **code** folder included with these instructions. Note that the `plotPost.R` and `HDIofMCMC.R` scripts are from Kruschke (2011). Therefore, you should set R's working directory to the `code` folder.

2 Data Description

Here, we will walk through how to analyze the actual bowhead whale data reported in the paper. However, instead of starting with the genotype data, we will instead start with those mark-recapture data already formatted as 1s and 0s: as mark-recapture histories. The rationale is that the earlier steps are only of interest to a small proportion of readers, whereas the included steps are relevant to any mark-recapture study.

3 Location-Specific Full Data Set (LS-FD)

3.1 Estimate Movement Rates

The first thing to do is estimate movement rates between all pairs of locations. Here we have data for four locations: Greenland, Igloolik, Pangnirtung, and Repulse Bay. The recaptures are shown in **Table 1**.

Table 1: Recaptures based on mark-recapture analysis of bowhead whale genetic data.

	Greenland	**Igloolik**	**Pangnirtung**	**Repulse Bay**
Greenland	5	0	2	0
Igloolik	-	23	6	0
Pangnirtung	-	-	3	3
Repulse Bay	-	-	-	1

The movement rates are estimated based on the proportion of recaptures within a location that were originally marked in the other compared location. These calculations are performed with the `movements.R` code, which can be referred to for further details. Briefly, this conducts a Bayesian hierarchical binomial analysis where the predicted variable (y_{ij}) is the number of recaptures across each considered pair of locations (i, j), the predictor variable (N_{ij}) is the total number of recaptures within each of the considered pair of locations, and the "probability of success" ($theta_{ij}$) is estimated across each pair of locations in a hierarchical manner. As equations, this is:

$$
\begin{aligned}
&\text{likelihood} && y_{ij} \sim binomial(N_{ij}, theta_{ij}) \\
&\text{priors} && theta_{ij} \sim normal(mean\_groups, sd\_groups) \\
&\text{hyperpriors} && mean\_groups \sim normal(0.5, 0.5) \\
& && sd\_groups \sim cauchy(0, 1)
\end{aligned}
$$

The result is a posterior probability distribution for each pairwise movement rate. The mean movement rate across all pairs of locations is then taken as the estimated movement rate to and from the unsampled location(s).

The `movements.R` code expects the data to be in a file containing one row for each pair of locations, and three columns. The first column contains the "label" for each pair of locations, the second column contains the number of recaptures *across* the considered locations, and the third column contains the total number of recaptures in both compared locations. These are stored in the "`ls-fd_movements.csv`" file in the **data** folder. It looks like this (from Table 1):

Pair	**Movers**	**Recaptures**
G-I	0	28
G-P	2	10
G-R	0	6
I-P	6	32
I-R	0	24
P-R	3	7

To run the code, we can `source` the code and provide the appropriate information. The `movements.R` function can accept seven arguments:

1. `nSites`: An integer indicating the number of sites containing data.
2. `filename`: the filename (and path) to the file containing the movement information.

3. `nIter`: How many steps to run the MCMC process.
4. `nChains`: The number of chains to use.
5. `print.diag`: Whether or not to print the diagnostics of the MCMC process (default = TRUE).
6. `plot.diag`: Whether or not to plot the diagnostics of the MCMC process (default = TRUE).
7. `plot.post`: Whether or not to plot the resulting posterior distributions (default = TRUE).

We can get these analyses to run by typing the lines below. **For this code to work as-written you must first set R's working directory to the `code` folder**.

```
source("movements.R")
movement.estimates = movements(nSites = 4, filename = "../data/ls-fd_movements.csv",
    nIter = 20000, nChains = 4)
write.csv(movement.estimates, "../results/ls-fd_movementEstimates.csv")
```

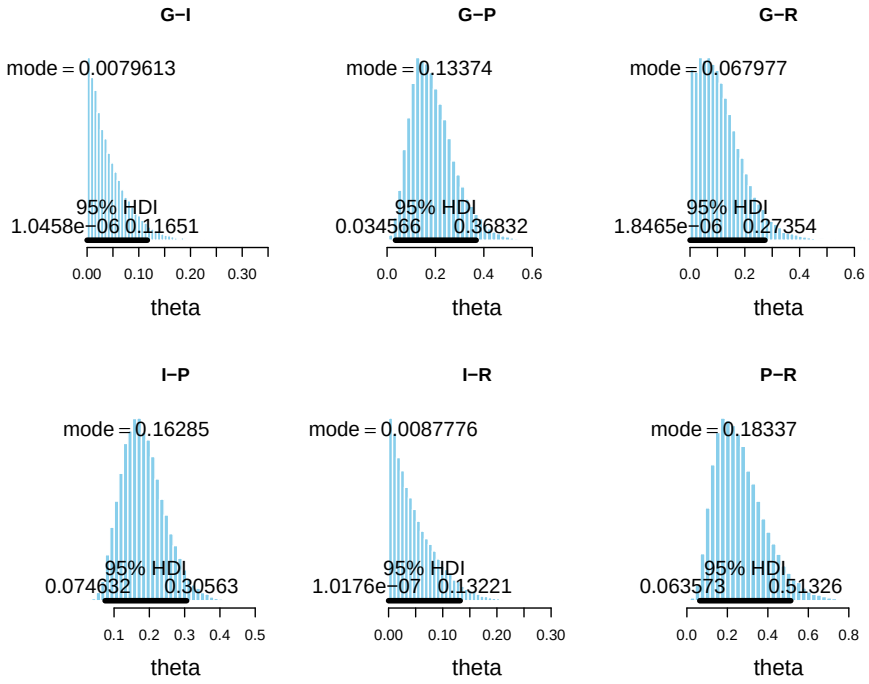
We can check the diagnostics, to ensure that the MCMC chains converged. First, we can look at the effective sample size and Rhat values.

	n_eff	Rhat
theta[1]	18272	1
theta[2]	19353	1
theta[3]	31230	1
theta[4]	17326	1
theta[5]	20840	1
theta[6]	13526	1
mean_groups	18754	1
sd_groups	10170	1

The function also allows the user to print the trace plots. However, those are not included in this document because the files are so large. The trace plots from this run, do, however, suggest convergence.

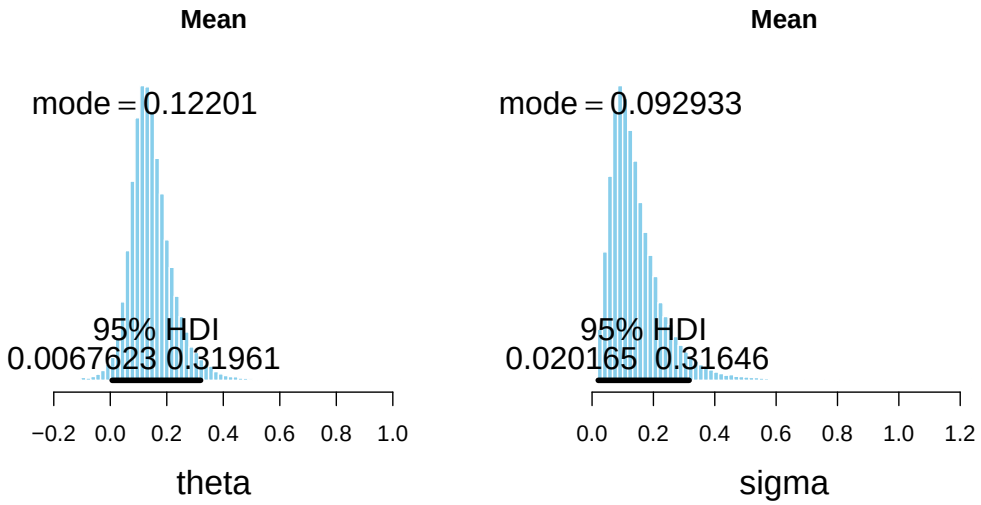
Then plot the results.

```
source("plotPost.R")
par(mfrow = c(2, 3))
movements = data.frame(movement.estimates)
histInfo = plotPost(movements$theta.1, xlab = "theta", main = "G-I", showMode = TRUE)
histInfo = plotPost(movements$theta.2, xlab = "theta", main = "G-P", showMode = TRUE)
histInfo = plotPost(movements$theta.3, xlab = "theta", main = "G-R", showMode = TRUE)
histInfo = plotPost(movements$theta.4, xlab = "theta", main = "I-P", showMode = TRUE)
histInfo = plotPost(movements$theta.5, xlab = "theta", main = "I-R", showMode = TRUE)
histInfo = plotPost(movements$theta.6, xlab = "theta", main = "P-R", showMode = TRUE)
```



Because these analyses were hierarchical, we also get an overall estimate for movement rates among locations, and an overall estimate for the standard deviation. The mean will be used as the estimate to and from unsampled location(s).

```
par(mfrow = c(1, 2))
histInfo = plotPost(movements$mean_groups, xlab = "theta", main = "Mean", showMode =
    TRUE)
histInfo = plotPost(movements$sd_groups, xlab = "sigma", main = "Mean", showMode = TRUE)
```



3.2 Estimate Abundance

This code uses a generic approach to estimate abundance in each location, assuming a closed population (e.g., Kéry & Schaub 2012). However, it does try to correct for unmarked individuals that may have moved into, or out of, the sampling area, as described in the paper. As a result, this code should be generic, and applicable to any other study trying to use this same approach.

3.2.1 Greenland

The "`Greenland2.csv`" file contains output from the `AlleleMatch` package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "`AlleleMatchConversion.R`" script, but first need to obtain some information about the file. This code is below.

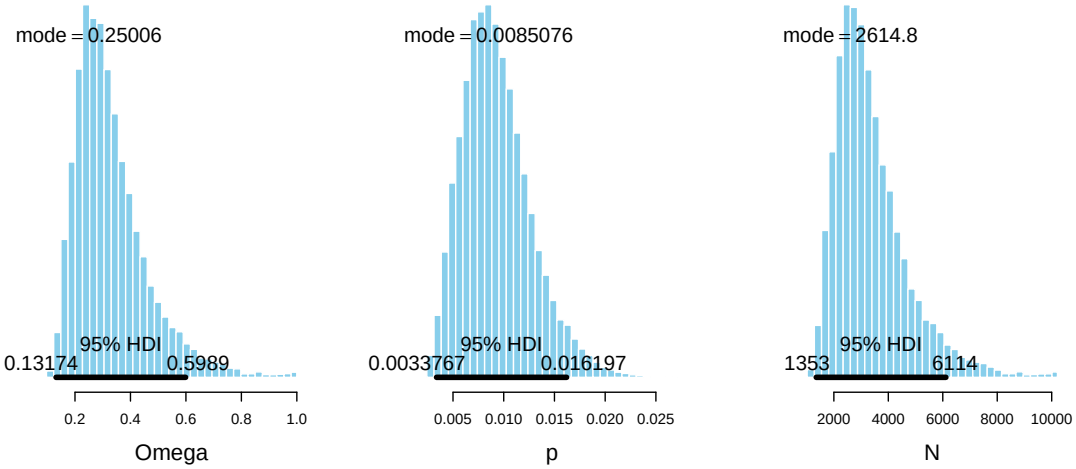
```
captures = read.table("../data/Greenland2.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "`abundance.R`" script to format and augment the data, and correct for missed recaptures. Note that here we are just running though the lines of code, rather than "sourcing" it. There are two parameters that the user will have to change: `m5post`", which is the mean movement rate estimate to and from the unsampled location(s); and `aug`" which is by how much the data should be augmented. **m5post** will always be 0.12201 for these analyses, because that is our estimate, and the Greenland data can be augmented by 10000.

Again, we can check model performance by checking the Rhat and trace plot values. The Rhat values are shown below, but the trace plots are not shown, although they looked great, and suggest that the chains converged on the same single estimate for each parameter. The parameters we are estimating are: omega - the "inclusion probability" or the probability with which a member of the augmented data set is included in the population of size N; p - the capture probability; and N - the abundance estimate.

	n_eff	Rhat
omega	4162	1
p	4610	1
N	4157	1

A plot of the posterior distribution for these estimates is below.



3.2.2 Igloolik

The "`Igloolic2.csv`" file contains output from the `AlleleMatch` package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "`AlleleMatchConversion.R`" script, but first need to obtain some information about the file. This code is below.

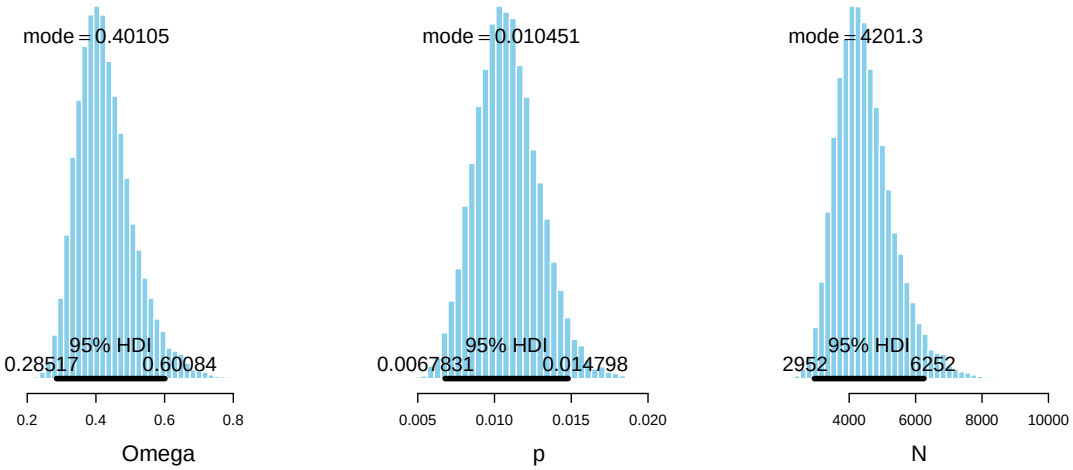
```
captures = read.table("../data/Igloolik2.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "`abundance.R`" script to format and augment the data, and correct for missed recaptures. Note that here we are just running though the lines of code, rather than "sourcing" it. There are two parameters that the user will have to change: `m5post`", which is the estimated movement rate estimate to and from the unsampled location(s); and `aug`" which is by how much the data should be augmented. **m5post** will always be 0.12201 for these analyses, because that is our estimate, and the Igloolik data can be augmented by 10,000.

Check model performance by checking the Rhat values and trace plot figures (trace plots not shown).

	n_eff	Rhat
omega	4833	1
p	5081	1
N	4833	1

A plot of the posterior distribution for these estimates is below.



3.2.3 Pangnirtung

The "`Pangnirtung2.csv`" file contains output from the `AlleleMatch` package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "`AlleleMatchConversion.R`" script, but first need to obtain some information about the file. This code is below.

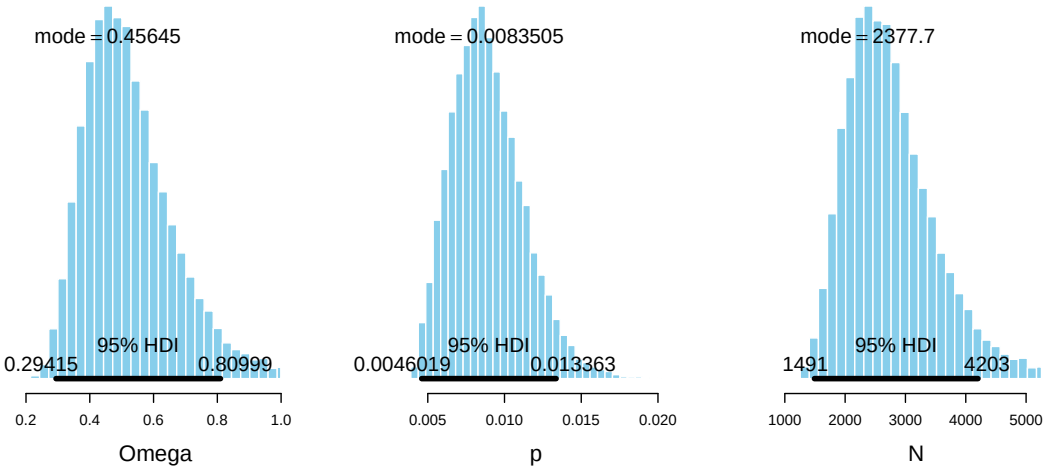
```
captures = read.table("../data/Pangnirtung2.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "`abundance.R`" script to format and augment the data, and correct for missed recaptures. Note that here we are just running though the lines of code, rather than "sourcing" it. There are two parameters that the user will have to change: `m5post`", which is the estimated movement rate estimate to and from the unsampled location(s); and `aug`" which is by how much the data should be augmented. **m5post** will always be 0.12201 for these analyses, because that is our estimate, and the Pangnirtung data can be augmented by 5000.

Check model performance by checking the Rhat values and trace plot figures (trace plots not shown).

	n_eff	Rhat
omega	4566	1
p	5268	1
N	4561	1

A plot of the posterior distribution for these estimates is below.



3.2.4 Repulse Bay

The "`RepulseBay2.csv`" file contains output from the `AlleleMatch` package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "`AlleleMatchConversion.R`" script, but first need to obtain some information about the file. This code is below.

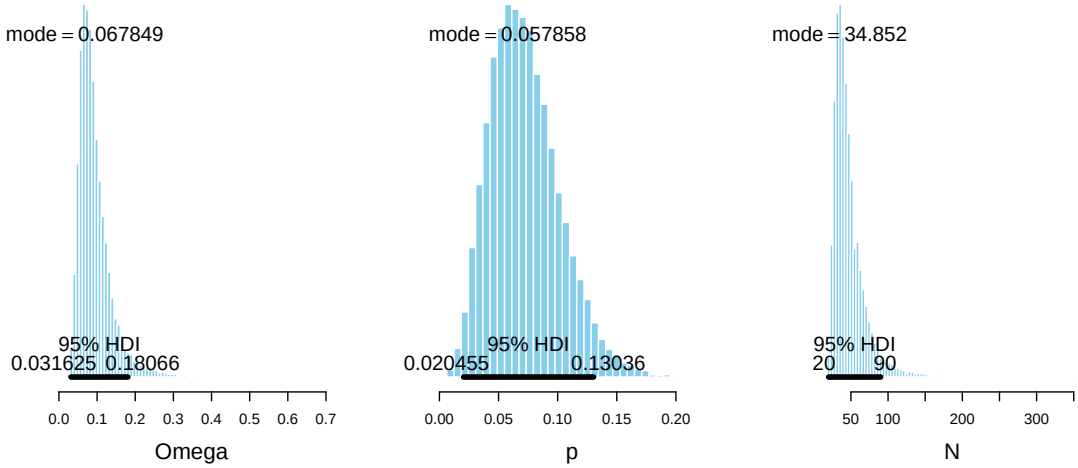
```
captures = read.table("../data/RepulseBay2.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "`abundance.R`" script to format and augment the data, and correct for missed recaptures. Note that here we are just running though the lines of code, rather than "sourcing" it. There are two parameters that the user will have to change: `m5post`", which is the estimated movement rate estimate to and from the unsampled location(s); and `aug`" which is by how much the data should be augmented. **m5post** will always be 0.12201 for these analyses, because that is our estimate, and the Pangnirtung data can be augmented by 500.

Check model performance by checking the Rhat values and trace plot figures (trace plots not shown).

	n_eff	Rhat
omega	5413	1
p	8145	1
N	5514	1

A plot of the posterior distribution for these estimates is below.



3.2.5 Missing Individuals

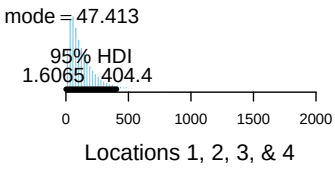
We can then use the "CellCalculation_4locations_Posteriors_1.R" script to obtain distributions for each row in the sighting histories. This requires all of the abundance and movement rate estimates calculated in earlier steps.

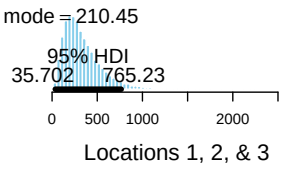
First, read all of the files into R.

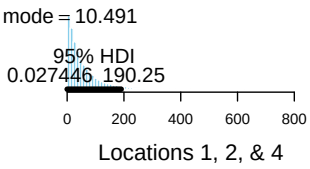
```
greenland = read.table("../results/ls-fd_Greenland.csv", header = TRUE, sep = ",")
igloolik = read.table("../results/ls-fd_Igloolik.csv", header = TRUE, sep = ",")
pangnirtung = read.table("../results/ls-fd_Pangnirtung.csv", header = TRUE, sep = ",
    ")
repulsebay = read.table("../results/ls-fd_RepulseBay.csv", header = TRUE, sep = ",")
movements = read.table("../results/ls-fd_movementEstimates.csv", header = TRUE, sep
    = ",")
```

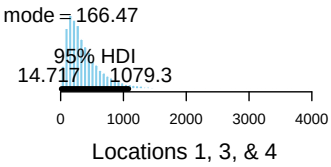
Then, load the script and run it using the appropriate data for each argument.

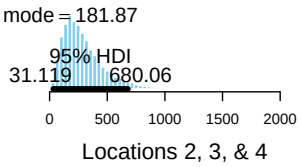
```
source("CellCalculation_4locations_Posteriors_1.R")
histcounts = cellcounts(pop1post = greenland[, "N"], pop2post = igloolik[, "N"],
    pop3post = pangnirtung[, "N"], pop4post = repulsebay[, "N"], m12post = movements
    [1:30000, "theta.1"], m13post = movements[1:30000, "theta.2"], m14post =
    movements[1:30000, "theta.3"], m23post = movements[1:30000, "theta.4"], m24post
    = movements[1:30000, "theta.5"], m34post = movements[1:30000, "theta.6"], m5post
     = movements[1:30000, "mean_groups"])
write.csv(histcounts, "../results/ls-fd_histcounts.csv")
```

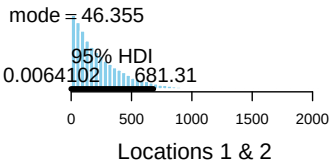


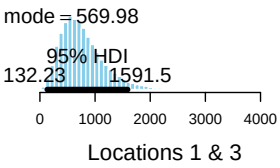


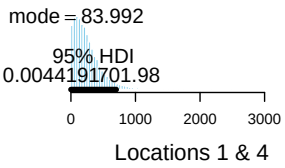


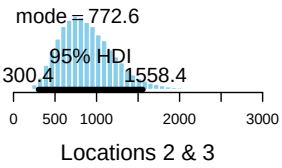


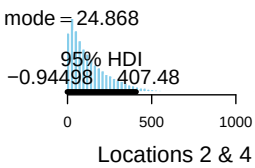


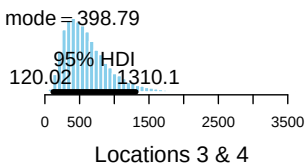


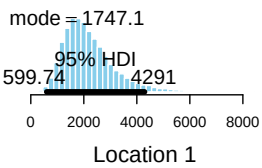


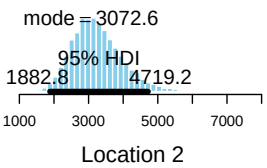


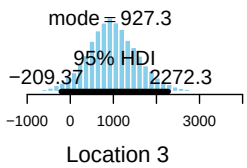


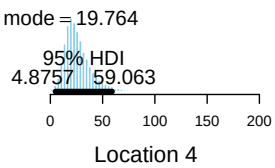












The results were as follows:

Sighting History	Num.	Green.	Igloo.	Pangn.	Repul.
1	47	YES	YES	YES	YES
2	210	YES	YES	YES	NO
3	10	YES	YES	NO	YES
4	166	YES	NO	YES	YES
5	182	NO	YES	YES	YES
6	46	YES	YES	NO	NO
7	570	YES	NO	YES	NO
8	84	YES	NO	NO	YES
9	773	NO	YES	YES	NO
10	25	NO	YES	NO	YES
11	399	NO	NO	YES	YES
12	1747	YES	NO	NO	NO
13	3073	NO	YES	NO	NO
14	927	NO	NO	YES	NO
15	20	NO	NO	NO	YES

Then we can estimate the number of missing individuals using the "`InferIndividuals_4locationsMetric_Counts.R`" script. Note that this script is set up specifically for four locations, and the code will therefore need to be changed if you wish to analyze data from a different number of locations.

This is the part where the values from the other sighting histories are used to predict the number of whales not "captured" in any sampled area. The predicted variable is the count of individuals with each sighting history. The predictor variables are each location, and individuals are coded as either "1" for "NO" or "2" for "YES" with respect to if they were not, or were, captured in each location. The regression equation used in this case is based on a negative binomial distribution, which is appropriate for count-based data when the mean and standard deviation may have different values (whereas a Poisson regression assumes the mean and standard deviation are equal):

$$mu = exp(\beta_0 + \beta_1 location_1 + \beta_2 location_2 + \beta_3 location_3 + \beta_4 location_4)$$

$$y \sim neg\_binomial\_2(mu, phi)$$

First, read the file with the posterior distributions of each sighting history into R (if it is not in R already).

```
histcounts = read.table("../results/ls-fd_histcounts.csv", header = TRUE, sep = ",")
```

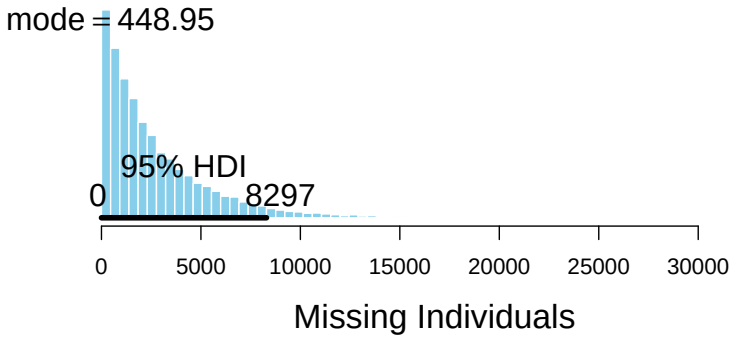
Then load and run the function to infer the number of individuals not sampled in any location. It does this by first using the counts of individuals with each sighting history to estimate the coefficients associated with each location (i.e., how being captured, or not, in each location and in combination with the other locations, impacts the count of individuals with that sighting history). Once these coefficients are estimated, they are used to predict the number of individuals not captured in any location (i.e., with a sighting history of 1, 1, 1, 1 across the four sampled locations). This function would take too long to run if the analyses were based on the entire posterior probability distributions generated for each sighting history (of which just the modes are shown in the table above). Instead, this function randomly samples 100 values from the posterior for each sighting history, and then bases the analyses on those. Even with this subsampling, this function still takes a little over two hours to run, using three processors on my 2013 Macbook Pro.

```
source("InferIndividuals_4locationsMetric_Counts.R")
missing = infermissing(histcounts = histcounts)
write.csv(missing, "../results/ls-fd_missing.csv")
```

Check model performance by checking the Rhat values and trace plot figures (trace plots not shown).

	n_eff	Rhat
b0	9415	1
b1[1]	9488	1
b1[2]	9482	1
b2[1]	10914	1
b2[2]	10908	1
b3[1]	9505	1
b3[2]	9523	1
b4[1]	10417	1
b4[2]	10418	1

A plot of the posterior distribution for these estimates is below.



The total population size can then be estimated, using the posterior probability distributions for all movement rates and abundance estimates. This can be done with the "CellCalculation_4locations_Posteriors_2.R" script. Again, this script is based on data for 4 observed locations, and will need to be changed use with data set with different characteristics. Note also that I had to truncate all posterior probabilities to 30,000 records, because they were originally of varying length.

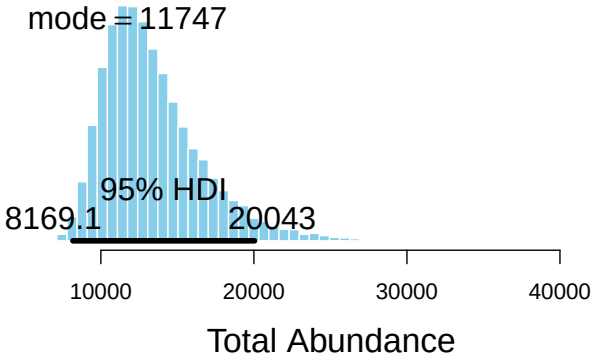
First, read the data into R, if they are not already there.

```
greenland = read.table("../results/ls-fd_Greenland.csv", header = TRUE, sep = ",")
igloolik = read.table("../results/ls-fd_Igloolik.csv", header = TRUE, sep = ",")
pangnirtung = read.table("../results/ls-fd_Pangnirtung.csv", header = TRUE, sep = ",
    ")
repulsebay = read.table("../results/ls-fd_RepulseBay.csv", header = TRUE, sep = ",")
movements = read.table("../results/ls-fd_movementEstimates.csv", header = TRUE, sep
    = ",")
missing = read.table("../results/ls-fd_missing.csv", header = TRUE, sep = ",")
```

Then, source the code and run.

```
source("CellCalculation_4locations_Posteriors_2.R")
```

```
totalPop = totals(pop1post = greenland[, "N"], pop2post = igloolik[, "N"], pop3post
    = pangnirtung[, "N"], pop4post = repulsebay[, "N"], pop5post = missing[, 2],
    m12post = movements[1:30000, "theta.1"], m13post = movements[1:30000, "theta.2"
    ], m14post = movements[1:30000, "theta.3"], m23post = movements[1:30000, "theta
    .4"], m24post = movements[1:30000, "theta.5"], m34post = movements[1:30000, "
    theta.6"], m5post = movements[1:30000, "mean_groups"])
write.csv(totalPop, "../results/ls-fd_TotalPop.csv")
```



4 Location-Specific 5-year Data Set (LS-5Y)

Note that for this time period there were not any recaptures in Repulse Bay, so all analyses are based on just the three locations of Greenland, Igloolik, and Pangnirtung.

4.1 Estimate Movement Rates

The number of recaptures with and among all sites for the 5-year data set are shown below.

	Greenland	**Igloolik**	**Pangnirtung**	**Repulse Bay**
Greenland	4	0	2	0
Igloolik	-	17	1	0
Pangnirtung	-	-	2	0
Repulse Bay	-	-	-	0

The movement rates are estimated based on the proportion of recaptures within a location that were originally marked in the other compared location. These calculations are performed with the `movements.R` code, as described above for the full data set.

The `movements.R` code expects the data to be in a file containing one row for each pair of locations, and three columns. The first column contains the "label" for each pair of locations, the second column contains the number of recaptures *across* the considered locations, and the third column contains the total number of recaptures in both compared locations. These are stored in the "`ls-5Y_movements.csv`" file in the **data** folder. It looks like this (from Table 1):

Pair	**Movers**	**Recaptures**
G-I	0	21
G-P	2	8
I-P	1	20

To run the code, we can `source` the code and provide the appropriate information, as described above. We can get these analyses to run by typing the lines below. **For this code to work as-written you must first set R's working directory to the `code` folder**.

```
source("movements.R")
movement.estimates = movements(nSites = 3, filename = "../data/ls-5Y_movements.csv",
    nIter = 20000, nChains = 4)
write.csv(movement.estimates, "../results/ls-5Y_movementEstimates.csv")
```

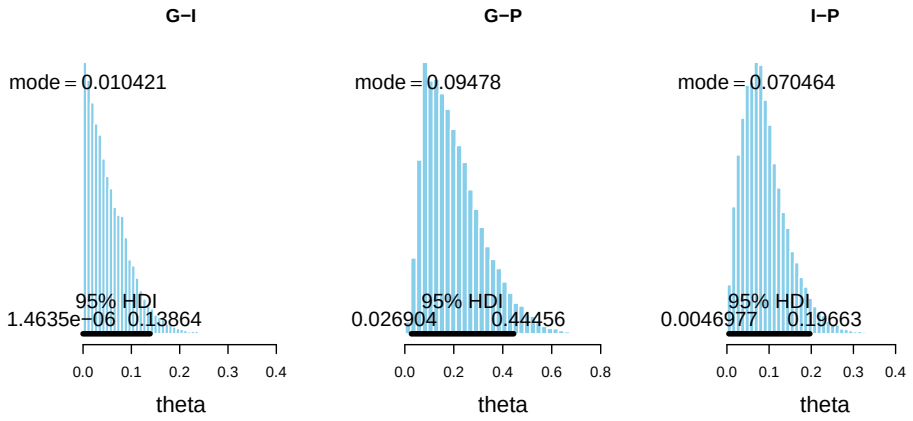
We can check the diagnostics, to ensure that the MCMC chains converged. First, we can look at the effective sample size and Rhat values.

	n_eff	Rhat
theta[1]	9260	1
theta[2]	6162	1
theta[3]	18922	1
mean_groups	11950	1
sd_groups	7669	1

The function also allows the user to print the trace plots. However, those are not included in this document because the files are so large. The trace plots from this run, do, however, suggest convergence.

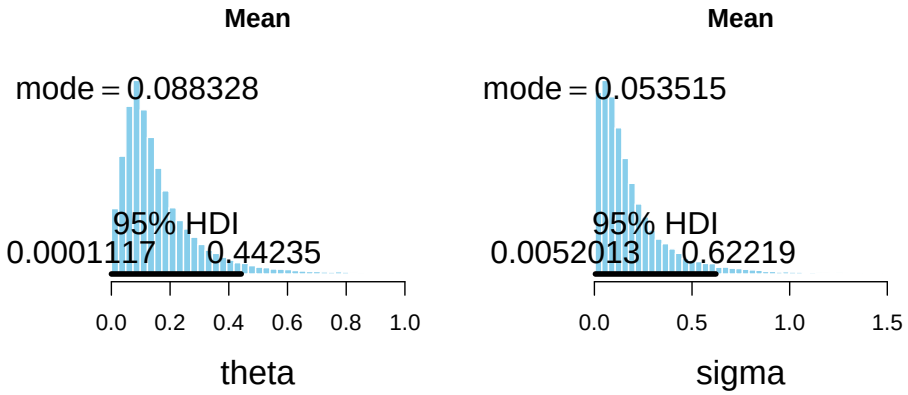
Then plot the results.

```
source("plotPost.R")
par(mfrow = c(1, 3))
movements = data.frame(movement.estimates)
histInfo = plotPost(movements$theta.1, xlab = "theta", main = "G-I", showMode = TRUE)
histInfo = plotPost(movements$theta.2, xlab = "theta", main = "G-P", showMode = TRUE)
histInfo = plotPost(movements$theta.3, xlab = "theta", main = "I-P", showMode = TRUE)
```



Because these analyses were hierarchical, we also get an overall estimate for movement rates among locations, and an overall estimate for the standard deviation. The mean will be used as the estimate to and from unsampled location(s).

```
par(mfrow = c(1, 2))
histInfo = plotPost(movements$mean_groups, xlab = "theta", main = "Mean", showMode =
    TRUE)
histInfo = plotPost(movements$sd_groups, xlab = "sigma", main = "Mean", showMode = TRUE,
     xlim = c(0, 1.5))
```



4.2 Estimate Abundance

4.2.1 Greenland

The "Greenland3.csv" file contains output from the AlleleMatch package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "AlleleMatchConversion.R" script, but first need to obtain some information about the file. This code is below.

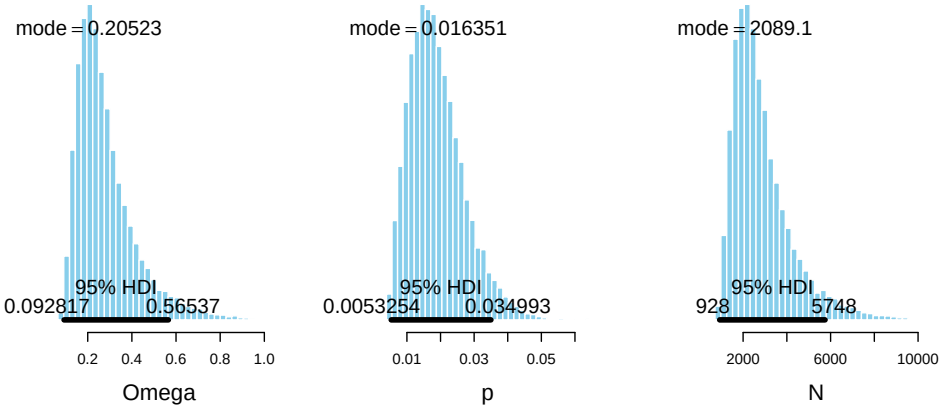
```
captures = read.table("../data/Greenland3.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "abundance.R" script to format and augment the data, and correct for missed recaptures. Note that here we are just running though the lines of code, rather than "sourcing" it. There are two parameters that the user will have to change: "m5post", which is the mean movement rate estimate to and from the unsampled location(s); and "aug" which is by how much the data should be augmented. **m5post** will always be 0.088328 for these analyses, because that is our estimate, and the Greenland data can be augmented by 10000.

Again, we can check model performance by checking the Rhat and trace plot values. The Rhat values are shown below, but the trace plots are not shown, although they looked great, and suggest that the chains converged on the same single estimate for each parameter. The parameters we are estimating are: omega - the "inclusion probability" or the probability with which a member of the augmented data set is included in the population of size N; p - the capture probability; and N - the abundance estimate.

	n_eff	Rhat
omega	4654	1
p	4988	1
N	4651	1

A plot of the posterior distribution for these estimates is below.



4.2.2 Igloolik

The "`Igloolik3.csv`" file contains output from the `AlleleMatch` package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "`AlleleMatchConversion.R`" script, but first need to obtain some information about the file. This code is below.

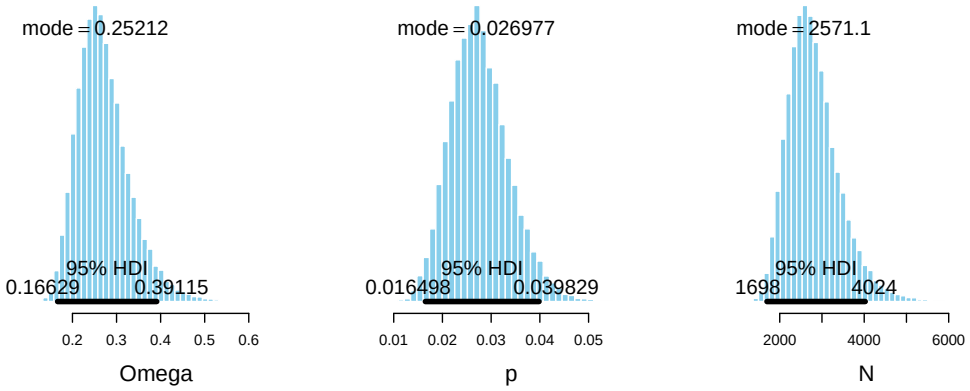
```
captures = read.table("../data/Igloolik3.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "`abundance.R`" script to format and augment the data, and correct for missed recaptures. Note that here we are just running though the lines of code, rather than "sourcing" it. There are two parameters that the user will have to change: `m5post`", which is the mean movement rate estimate to and from the unsampled location(s); and `aug`" which is by how much the data should be augmented. **m5post** will always be 0.088328 for these analyses, because that is our estimate, and the Igloolik data can be augmented by 10000.

Again, we can check model performance by checking the Rhat and trace plot values. The Rhat values are shown below, but the trace plots are not shown, although they looked great, and suggest that the chains converged on the same single estimate for each parameter. The parameters we are estimating are: omega - the "inclusion probability" or the probability with which a member of the augmented data set is included in the population of size N; p - the capture probability; and N - the abundance estimate.

	n_eff	Rhat
omega	5144	1
p	5330	1
N	5135	1

A plot of the posterior distribution for these estimates is below.



4.2.3 Pangnirtung

The "Pangnirtung3.csv" file contains output from the AlleleMatch package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "AlleleMatchConversion.R" script, but first need to obtain some information about the file. This code is below.

```
captures = read.table("../data/Pangnirtung3.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

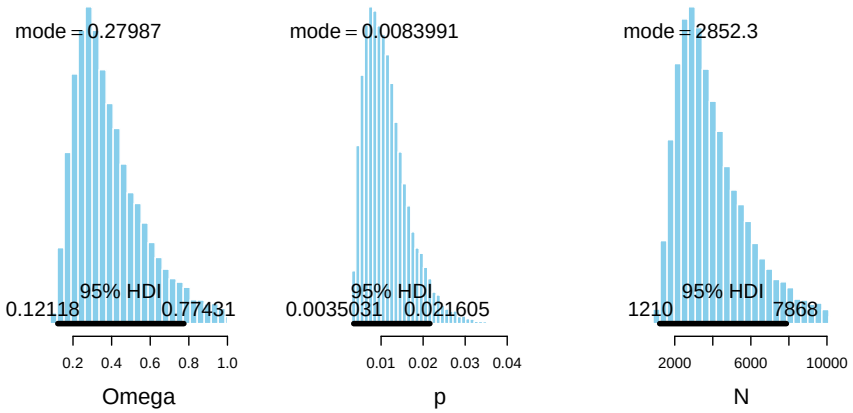
We can then use the code from the "abundance.R" script to format and augment the data, and correct for missed recaptures. Note that here we are just running though the lines of code, rather than "sourcing" it. There are two parameters that the user will have to change: m5post", which is the mean movement rate estimate to and from the unsampled location(s); and aug" which is by how much the data should be augmented. **m5post** will always be 0.088328 for these analyses, because that is our estimate, and the Pangnirtung data can be augmented by 10000.

Again, we can check model performance by checking the Rhat and trace plot values. The Rhat values are shown below, but the trace plots are not shown, although they looked great, and suggest that the chains

converged on the same single estimate for each parameter. The parameters we are estimating are: omega - the "inclusion probability" or the probability with which a member of the augmented data set is included in the population of size N; p - the capture probability; and N - the abundance estimate.

	n_eff	Rhat
omega	4769	1
p	5059	1
N	4773	1

A plot of the posterior distribution for these estimates is below.



4.2.4 Missing Individuals

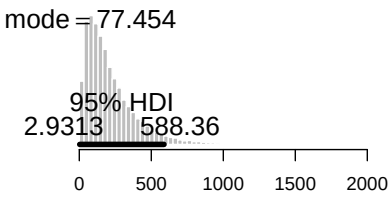
Now we have just three locations, rather than four, and therefore have to use a different script for estimating the number of individuals with each sighting history. Specifically, we can then use the "CellCalculation_3locations_Posteri script to obtain distributions for each row in the sighting histories. This requires all of the abundance and movement rate estimates calculated in earlier steps.

First, read all of the files into R.

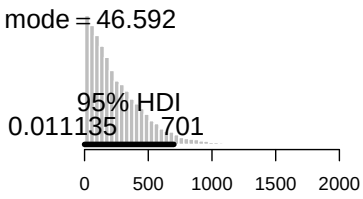
```
greenland = read.table("../results/ls-5Y_Greenland.csv", header = TRUE, sep = ",")
igloolik = read.table("../results/ls-5Y_Igloolik.csv", header = TRUE, sep = ",")
pangnirtung = read.table("../results/ls-5Y_Pangnirtung.csv", header = TRUE, sep = ",
    ")
movements = read.table("../results/ls-5Y_movementEstimates.csv", header = TRUE, sep
    = ",")
```

Then, load the script and run it using the appropriate data for each argument.

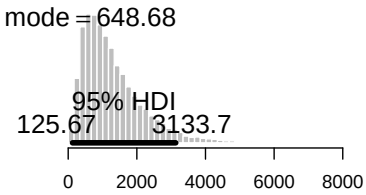
```
source("CellCalculation_3locations_Posteriors_1.R")
histcounts = cellcounts(pop1post = greenland[, "N"], pop2post = igloolik[, "N"],
    pop3post = pangnirtung[, "N"], m12post = movements[1:30000, "theta.1"], m13post
    = movements[1:30000, "theta.2"], m23post = movements[1:30000, "theta.3"], m5post
     = movements[1:30000, "mean_groups"])
write.csv(histcounts, "../results/ls-5Y_histcounts.csv")
```



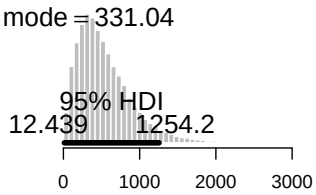
Locations 1, 2, & 3



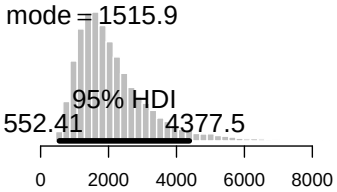
Locations 1 & 2



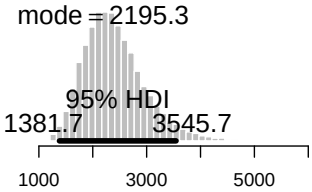
Locations 1 & 3



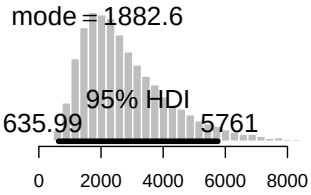
Locations 2 & 3



Location 1



Location 2



Location 3

The results were as follows:

Sighting History	Num.	Green.	Igloo.	Pangn.
1	77	YES	YES	YES
2	47	YES	YES	NO
3	649	YES	NO	YES
4	331	NO	YES	YES
5	1516	YES	NO	NO
6	2195	NO	YES	NO
7	1883	NO	NO	YES

Then we can estimate the number of missing individuals using the "`InferIndividuals_3locationsMetric_Counts.R`" script. Note that this script is set up specifically for three locations, and the code will therefore need to be changed if you wish to analyze data from a different number of locations.

This is the part where the values from the other sighting histories are used to predict the number of whales not "captured" in any sampled area. The predicted variable is the count of individuals with each sighting history. The predictor variables are each location, and individuals are coded as either "1" for "NO" or "2" for "YES" with respect to if they were not, or were, captured in each location. The regression equation used in this case is based on a negative binomial distribution, which is appropriate for count-based data when the mean and standard deviation may have different values (whereas a Poisson regression assumes the mean and standard deviation are equal):

$$mu = exp(\beta_0 + \beta_1 location_1 + \beta_2 location_2 + \beta_3 location_3$$

$$y \sim neg\_binomial\_2(mu, phi)$$

First, read the file with the posterior distributions of each sighting history into R (if it is not in R already).

```
histcounts = read.table("../results/ls-5Y_histcounts.csv", header = TRUE, sep = ",")
```

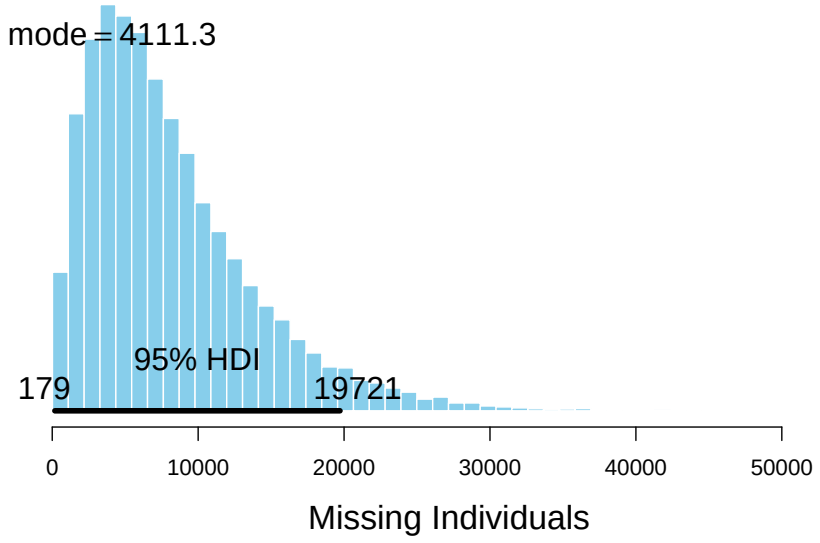
Then load and run the function to infer the number of individuals not sampled in any location. It does this by first using the counts of individuals with each sighting history to estimate the coefficients associated with each location (i.e., how being captured, or not, in each location and in combination with the other locations, impacts the count of individuals with that sighting history). Once these coefficients are estimated, they are used to predict the number of individuals not captured in any location (i.e., with a sighting history of 1, 1, 1 across the three sampled locations). This function would take too long to run if the analyses were based on the entire posterior probability distributions generated for each sighting history (of which just the modes are shown in the table above). Instead, this function randomly samples 100 values from the posterior for each sighting history, and then bases the analyses on those. Even with this subsampling, this function still takes a little over two hours to run, using three processors on my 2013 Macbook Pro.

```
source("InferIndividuals_3locationsMetric_Counts.R")
missing = infermissing(histcounts = histcounts)
write.csv(missing, "../results/ls-5Y_missing.csv")
```

Check model performance by checking the Rhat values and trace plot figures (trace plots not shown).

	n_eff	Rhat
b0	7543	1
b1[1]	10964	1
b1[2]	10967	1
b2[1]	6854	1
b2[2]	6856	1
b3[1]	6653	1
b3[2]	6657	1

A plot of the posterior distribution for these estimates is below.



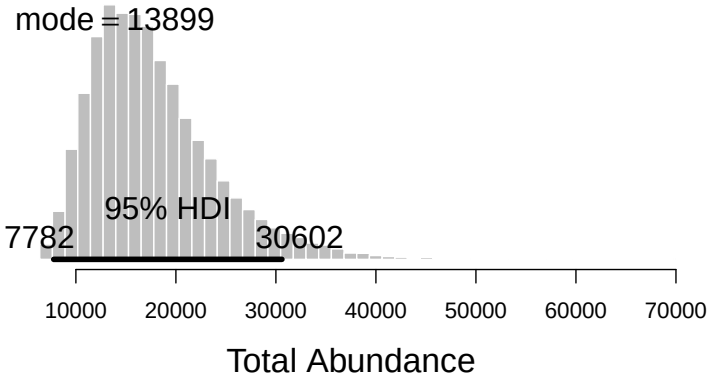
The total population size can then be estimated, using the posterior probability distributions for all movement rates and abundance estimates. This can be done with the "CellCalculation_3locations_Posteriors_2.R" script. Again, this script is based on data for 4 observed locations, and will need to be changed use with data set with different characteristics. Note also that I had to truncate all posterior probabilities to 30,000 records, because they were originally of varying length.

First, read the data into R, if they are not already there.

```
greenland = read.table("../results/ls-5Y_Greenland.csv", header = TRUE, sep = ",")
igloolik = read.table("../results/ls-5Y_Igloolik.csv", header = TRUE, sep = ",")
pangnirtung = read.table("../results/ls-5Y_Pangnirtung.csv", header = TRUE, sep = ",
    ")
movements = read.table("../results/ls-5Y_movementEstimates.csv", header = TRUE, sep
    = ",")
missing = read.table("../results/ls-5Y_missing.csv", header = TRUE, sep = ",")
```

Then, source the code and run.

```
source("CellCalculation_3locations_Posteriors_2.R")
totalPop = totals(pop1post = greenland[, "N"], pop2post = igloolik[, "N"], pop3post
    = pangnirtung[, "N"], pop5post = missing[, 2], m12post = movements[1:30000, "
    theta.1"], m13post = movements[1:30000, "theta.2"], m23post = movements[1:30000,
     "theta.3"], m5post = movements[1:30000, "mean_groups"])
write.csv(totalPop, "../results/ls-5Y_TotalPop.csv")
```



5 Location-Independent Full Data Set (LI-FD)

Another approach is that we can ignore the location information and treat the entire data set as one large mark-recapture data set. To do this, we can use a subset of the same commands and scripts as before. For the location-independent full data set, the file is called "`li-fd_totaldata.csv`". This file contains output from the `AlleleMatch` package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "`AlleleMatchConversion.R`" script, but first need to obtain some information about the file. This code is below.

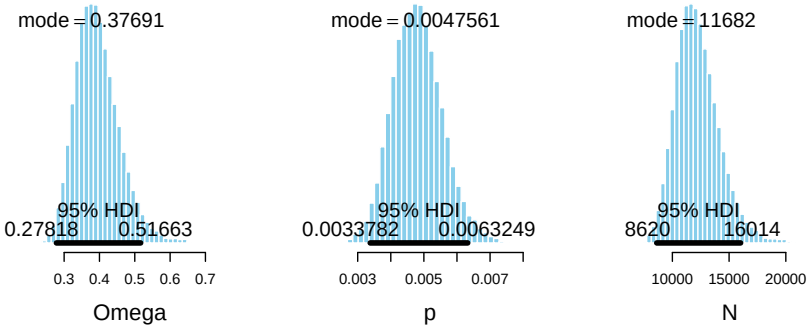
```
captures = read.table("../data/li-fd_totaldata.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "`abundance_Single.R`" script to format and augment the data. Note that here we are just running though the lines of code, rather than "sourcing" it. There is one parameter that the user will have to change: `aug`" which is by how much the data should be augmented. The total data set here can be augmented by 30000.

Again, we can check model performance by checking the Rhat and trace plot values. The Rhat values are shown below, but the trace plots are not shown, although they looked great, and suggest that the chains converged on the same single estimate for each parameter. The parameters we are estimating are: omega - the "inclusion probability" or the probability with which a member of the augmented data set is included in the population of size N; p - the capture probability; and N - the abundance estimate.

	n_eff	Rhat
omega	4435	1
p	4679	1
N	4435	1

A plot of the posterior distribution for these estimates is below.


mode = 0.37691
95% HDI
0.27818
0.51663
0.3
0.4
0.5
0.6
0.7
Omega
mode = 0.0047561
95% HDI
0.0033782
0.0063249
0.003
0.005
0.007
p
mode = 11682
95% HDI
8620
16014
10000
15000
20000
N

6 Location-Independent 5-Year Data Set (LI-5Y)

For the location-independent 5-year data set, the file is called "`li-5Y_totaldata.csv`". This file contains output from the `AlleleMatch` package. It therefore needs to be converted into a recapture matrix of 0s and 1s. We will do this using the "`AlleleMatchConversion.R`" script, but first need to obtain some information about the file. This code is below.

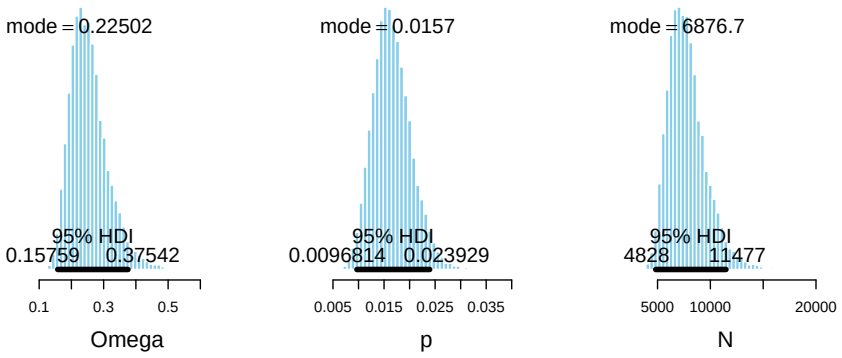
```
captures = read.table("../data/li-5Y_totaldata.csv", header = TRUE, sep = ",")

ninds = length(unique(captures$original))
nlines = length(captures$original)

l1 = sort(unique(captures[, 3]))
l2 = sort(unique(captures[, 6]))
l3 = c(l1, l2)
periods = sort(unique(l3))
nperiods = length(periods)

source("AlleleMatchConversion.R")
pophist = histconversion(captures = captures, ninds = ninds, nlines = nlines,
    periods = periods, nperiods = nperiods)
```

We can then use the code from the "`abundance_Single.R`" script to format and augment the data. Note that here we are just running though the lines of code, rather than "sourcing" it. There is one parameter that the user will have to change: `aug`" which is by how much the data should be augmented. The total data set here can be augmented by 30000.

Again, we can check model performance by checking the Rhat and trace plot values. The Rhat values are shown below, but the trace plots are not shown, although they looked great, and suggest that the chains converged on the same single estimate for each parameter. The parameters we are estimating are: omega - the "inclusion probability" or the probability with which a member of the augmented data set is included in the population of size N; p - the capture probability; and N - the abundance estimate.

	n_eff	Rhat
omega	4384	1
p	4680	1
N	4381	1

A plot of the posterior distribution for these estimates is below.



7 References

Kéry M, Schaub M (2012) *Bayesian Population Analysis Using WinBUGS: A Hierarchical Perspective.* Academic Press, Burlington, MA.

Kruschke JK (2011) *Doing Bayesian Data Analysis: A Tutorial With R and BUGS.* Academic Press, Burlington, MA.