Creating Migratory Networks in R: mignette

Matt DeSaix

2023-08-20

Contents

1 Introduction **5**
1.1 Installation 6

2 Quick start example **7**

3 Breeding nodes **11**
3.1 ebirdst 11
3.2 Creating the genoscape 12

4 Nonbreeding nodes **15**
4.1 Subsetting ecoregions 15
4.2 Select sampled regions 16

5 Node assignment **19**

6 Relative Abundance **21**
6.1 Get relative abundance for a genoscape (raster) 21
6.2 Get relative abundance for a vector 22
6.3 Combine the abundance data 22

7 Migratory Network Model **23**
7.1 Input data 23
7.2 Migratory network model (JAGS) 24

8 Network visualization **27**
8.1 Other visualization options 29

Chapter 1

Introduction

This vignette outlines how to create migratory networks using the R package `mignette` (**mig**ratory **net**work **t**ools **e**nsemble). `mignette` was developed to facilitate conservation and management decision-making for migratory wildlife populations using migratory networks [Ruegg et al., 2020, Taylor and Norris, 2010]. The core function of `mignette` is to model the connectivity of populations across different stages of the annual cycle. To model these migratory networks in `mignette`, users need to provide:

1. The number of individuals connecting migratory populations from two stages of the annual cycle
2. Relative abundance data for the populations

We designed `mignette` specifically for use with population assignment data from genetic markers [Ruegg et al., 2014, DeSaix et al., 2019], and we use genetic markers to delineate breeding populations (i.e., nodes in the migratory network model). We have since extended `mignette` to be able to model migratory networks that include geolocator data for assignment. While it is outside the scope of `mignette` for providing an exhaustive tool set of methods to delineate these nodes, given our focus on the use of genetic data, we provide examples and resources for users to delineate breeding populations from genetic data. For nonbreeding populations, we delineate these spatially by *ecoregions* ADD LINK TO ECOREGION DATA SOURCE.

We anticipate users of `mignette` to be interested in modelling migratory networks for a wide-range of taxa, therefore users can provide relative abundance information from any data source. However, we also provide information on obtaining relative abundance for bird data from eBird and have developed tools in `mignette` to facilitate the calculation of relative abundance values.

For an introductory demonstration of creating a migratory network with `mignette`, see the quick start example. Subsequent chapters of the vignette

provide more detailed instructions and examples for delineating nodes (populations in the network model), calculating relative abundance for nodes, and running and visualizing the network model.

1.1 Installation

You can install the development version of `mignette` from GitHub with:

```
# install.packages("remotes")
remotes::install_github("mgdesaix/mignette")
```

The migratory network model will be run using R packages that run JAGS (Just Another Gibbs Sampler). JAGS is a specific software for conducting analysis of Bayesian hierarchical models using Markov Chain Monte Carlo simulation. JAGS can be downloaded here

The primary packages for this vignette are:

```
library(mignette)
library(sf)
library(terra)
library(tidyverse)
library(ebirdst) # for avian abundance data from eBird Status and Trends
library(rjags) # for network model
library(jagsUI) # for network model
library(ggnewscale) # for network visualization
library(tidyterra) # for plotting terra objects with ggplot
```

As `mignette` makes use of the `terra` package, all raster and vector (e.g., polygons) are provided as *SpatRaster* and *SpatVector* objects. See the `terra` package for more details on these types of objects in R.

Chapter 2

Quick start example

Here, we demonstrate a quick example of how to create a migratory network when the user has all of the data required. To run this tutorial, load the following packages:

```
library(tidyverse)
library(mignette)
library(rjags)
library(jagsUI)
library(ggnewscale)
```

The data required are:

- Relative abundance matrix for each node
- Assignment matrix of individuals among nodes

We provide an example of these data in `mignette`, with assignment data from 3 populations from the breeding range (WB = Western Boreal, NT = Northern Temperate, ST = Southern Temperate) and nonbreeding range (ALM = Atlantic Lowland Mexico, CAR = Caribbean, AONU = Amazon/Orinoco-Northern Uplands) of the American Redstart (*Setophaga ruticilla*). The assignment matrix specifies the number of individuals that have been sampled or detected that migrate between different populations (i.e. *connect* the nodes).

```
mignette::amre_assign
```

Breeding	CAR	AONU	ALM	HCA	LCA
BR	2	0	0	0	0
MP	0	9	0	0	0
NT	54	0	3	0	0
ST	12	12	0	4	1
WB	1	0	19	1	13

Assignment data input into `mignette` needs to follow the above format, where the first column specifies breeding population IDs while subsequent columns are the nonbreeding populations.

We also provide the relative abundance of these populations:

```
mignette::amre_abundance
```

Population	Relative_abundance
BR	2403
ST	9419
MP	19011
NT	72147
WB	26080
HCA	326
AONU	1139
LCA	2802
ALM	3169
CAR	7987

The *relative abundance* data needs to follow the above format for input into `mignette` functions with population IDs (same names as in the *assignment* file) in the first column and relative abundance values in the second column. Column names can follow any naming convention when inputting these data into `mignette`.

For the following functions, we specify the order of the populations we are using for the model. Here, we are just ordering populations geographically by longitude to facilitate straightforward interpretation of the output.

```
bnode_names <- c("WB", "BR", "NT", "ST", "MP")
wnode_names <- c("ALM", "LCA", "HCA", "CAR", "AONU")
```

The following code provides the necessary data to run the JAGS model. To create the migratory network, the user first creates a text file specifying the JAGS model to be used, providing the name of the file to be saved (`base_filename`) and the type of model type (`model_type`). Currently `mignette` supports two model types based on the type of data used to determine assignment of individuals: `1` indicates that only genetic data were used for assignment, and `2` indicates that there's assignment data from both genetic and geolocator data. Here, the example only uses genetic data. `get_jags_model()` saves a `.txt` file with the `base_filename` and stores that name as a variable for use in JAGS. We also specify the desired order of the breeding populations (`bnode_names`) and the nonbreeding populations (`wnode_names`). Finally, we use these as input into the function `get_jags_data()` to prepare the data appropriately for the model.

```
base_filename <- mignette::get_jags_model(base_filename = "amre.genetic.model", model_t
jags_data <- mignette::get_jags_data(abundance = mignette::amre_abundance,
                                     assignment = mignette::amre_assign,
```

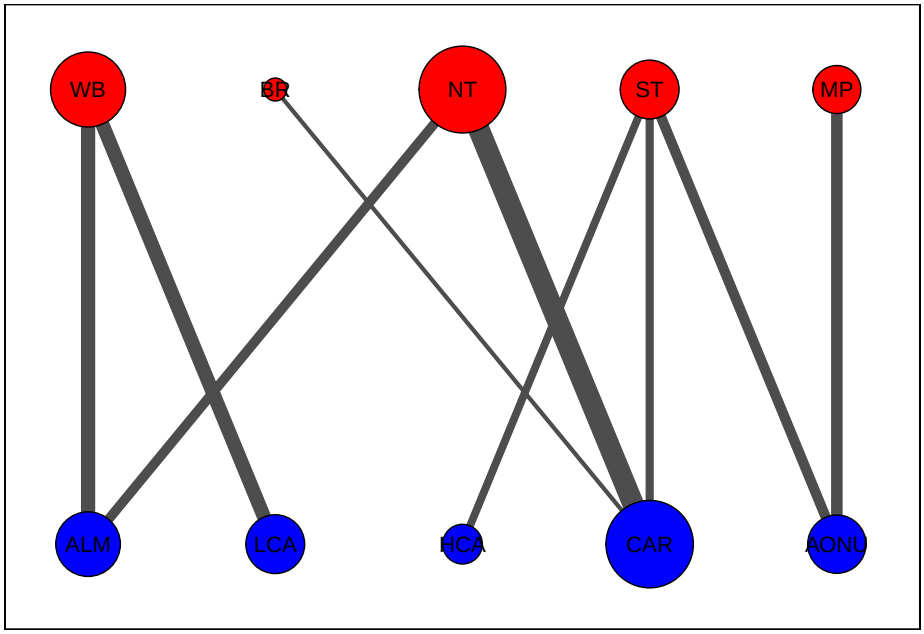
```
                              bnode_names = bnode_names,
                              wnode_names = wnode_names)
```

Now the user can use the output of `jags_data` into JAGS to run the actual model:

```
parameters <- c("conn_g")
ni <- 500000
nt <- 4
nb <- 100000
nc <- 2
jags_out <- autojags(jags_data, inits=NULL, parameters, paste0(base_filename,".txt"),
                     n.chains = nc, n.thin =nt, iter.increment=ni,
                     max.iter = ni*50+nb, n.burnin = nb,
                     n.adapt= NULL, parallel=TRUE)
amre_conn <- jags_out$mean$conn_g
```

The connectivity between the nodes is provided by the `conn_g` parameter of the model, which is accessed from the above code. We provide this output for the example as well with `mignette::amre_conn`. We also provide functions for basic visualization of the network. A threshold (`connected_tol`) can be set to remove very weak connectivity for better visualizing the network as shown below.

```
# set threshold for not visualizing minimal connectivity
net <- mignette::net_create(mignette::amre_conn,
                            node.names = list(bnode_names, wnode_names),
                            connected_tol = 0.02)
#set the display size range for nodes (min and max), default 1-10
net$display_par$node_size_scale<-c(5,20)
#set the display size range for edges (min and max), default 1-10
net$display_par$edge_size_scale<-c(1,5)
plot(mignette::net_draw(net))
```



In this visualization, node size corresponds to the amount of connectivity with that population and edge size corresponds to the amount of connectivity between the populations. By default, breeding populations are in the top row (red) and nonbreeding/wintering populations are in the bottom row (blue).

This sums up the basics of creating and visualizing a migratory network. We encourage users to explore and build upon the visualization tools we provide (e.g. overlay the migratory networks on geographic ranges) - the options are endless, enjoy!

Chapter 3

Breeding nodes

Delineating breeding nodes is necessary for our migratory network model for both 1) assignment among populations, and 2) specifying a region for relative abundance. Here, we show how breeding nodes can be delineated by genetically distinct populations on the breeding grounds. In this example, we'll show how to use eBird Status and Trends data to specify the breeding range and then use genetic data from admixture analyses to specify the spatial extent of the breeding nodes.

3.1 ebirdst

In our migratory network analyses, the eBird Status and Trends data is used to delineate the different stages of the annual cycle. Prior to doing anything with eBird Status and Trends data, you will need to download the `ebirdst` package, and then get access to the data. **You will need to follow the most up-to-date instructions from the `ebirdst` developers for getting the abundance data. Currently, that information is here: https://ebird.github.io/ebirdst/**

To download the package:

```
# install.packages("remotes")
remotes::install_github("CornellLabofOrnithology/ebirdst")
```

Then, get access to `ebirdst` data at https://ebird.org/st/request. You will receive a key to download `ebirdst` data and you can enter that key in R:

```
ebirdst::set_ebirdst_access_key("XXXXX")
```

where `"XXXXX"` is the key.

By following instructions from the `ebirdst` developers, you can obtain polygons

of the breeding and nonbreeding ranges of avian species (see https://ebird.github.io/ebirdst/ for details).

3.2 Creating the genoscape

A genoscape is the collection of genetically distinct populations that make up a species' range [Ruegg et al., 2021]. Typically, for migratory species, the genoscape describes this population structure on the breeding range because the nonbreeding populations can can contain individuals from different breeding populations.

We will outline the main genoscape creation steps here, but full instructions on creating a genoscape map can be found in Eric Anderson's Github project Make a Bird Genoscape Project map. The input data needed for a genoscape are:

- Individual Q-value matrix
- Lat/lon matrix of individuals
- Breeding range polygon

The Q-value matrix is obtained from individual admixture analyses (e.g. Structure, Admixture, `snmf` fuction from the LEA R-package). Latitude/longitude coordinates are for the individual samples used in the Q-value matrix. Breeding range polygons can be obtained from `ebirdst` (see previous section).

The `amre_breeding_data` data set provides admixture results (Q-values) for five genetic clusters for American Redstart (DeSaix et al. 2023) and metadata for the sampled individuals.

```
Q_matrix <- mignette::amre_breeding_data %>%
  dplyr::select(WB, BR, NT, ST, MP) %>%
  as.matrix()

coords <- mignette::amre_breeding_data %>%
  dplyr::select(Lon, Lat) %>%
  as.matrix()
```

The larger data objects in `mignette` are stored in `extdata`:

```
breeding_range <- system.file("extdata", "amre_breeding_range.Rds", package = "mignette
  terra::vect()

breeding_range_st <- sf::st_as_sf(mignette::amre_breeding_range) # tess3Q_map_rasters
```

We will also specify custom colors to correspond to the published genoscape.

```
cluster_colors <-  c(
  `ST` = "#FF99FF", # pink/Southern Temperate
  `BR` = "#3399FF", # blue/Basin Rockies
  `NT` = "#FFFF33", # yellow/Northern Temperate
```

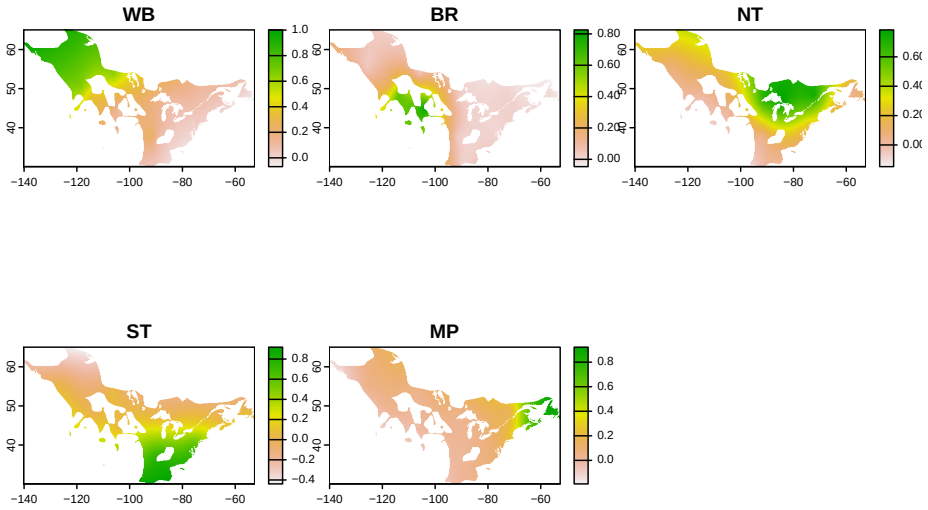
```
  `WB` = "#339933", # green/Western Boreal
  `MP` = "#CC0033" # red/Maritime Provinces
)
```

We will use a modified version of the `tess3r` package to create the genoscape rasters.

```
# remotes::install_github("eriqande/TESS3_encho_sen")
amre_genoscape <- tess3r::tess3Q_map_rasters(
  x = Q_matrix,
  coord = coords,
  map.polygon = breeding_range_st,
  window = sf::st_bbox(breeding_range_st),
  resolution = c(300,300), # if you want more cells in your raster, set higher
  col.palette = tess3r::CreatePalette(cluster_colors, length(cluster_colors)),
  method = "map.max",
  interpol = tess3r::FieldsKrigModel(10),
  main = "Ancestry coefficients",
  xlab = "Longitude",
  ylab = "Latitude",
  cex = .4
)
names(amre_genoscape) <- colnames(Q_matrix)
amre_genoscape <- terra::rast(amre_genoscape) # convert from rasterbrick to spatRaster
crs(amre_genoscape) <- "EPSG:4326" # Set CRS projection
```

Check out the resulting genoscape:

```
plot(amre_genoscape)
```



STOP

The rasters for the genoscape are all that are needed for obtaining information on relative abundance for the different populations. You can continue on to the relative abundance chapter if you are ready to do that with the genoscape. Or if you still need to create the wintering nodes, check out the next chapter on wintering nodes. The following section is not necessary for the migratory network but details how to covert genoscape rasters to polygons if the `mignette` user is interested in doing so.

3.2.1 Genoscape polygons

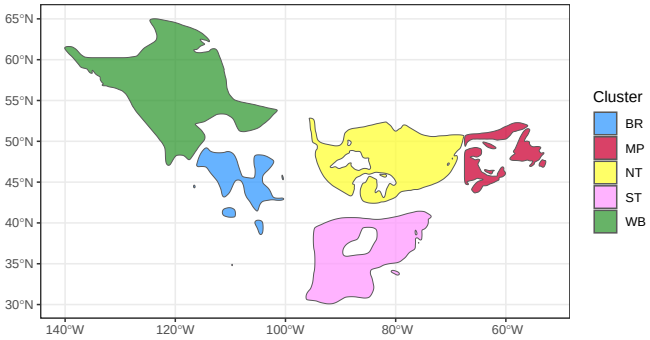
Using the genoscape rasters we will convert them to polygons, using the handy `scape_to_shape()` function. The `prob_threshold` parameter specifies the value to determine if a raster cell is included in the polygon for that genoscape. This value should be customized for different species to check for overlap of genoscape polygons, which is not desirable. Setting too high of a threshold will create very small breeding nodes, while too low of a threshold will result in large, overlapping breeding nodes.

Check out the polygons

```
genoscape_polygons <- mignette::scape_to_shape(x = amre_genoscape,
                                               prob_threshold = 0.5)
```

```
## Spherical geometry (s2) switched off
```

```
ggplot() +
  geom_sf(data = genoscape_polygons,alpha = 0.75, aes(fill = Cluster)) +
  scale_fill_manual(values = cluster_colors) +
  theme_bw()
```



Note that the lower population membership probabilities occur at the boundaries of different populations, thereby making the polygons discontinuous compared to the entire range

Now on to creating the wintering nodes

Chapter 4

Nonbreeding nodes

For the migratory networks, we will use ecoregions to define the nonbreeding nodes. However, other nonbreeding nodes could be defined by the user instead. If you already have polygons defining your nodes interest, then continue to the relative abundance section.

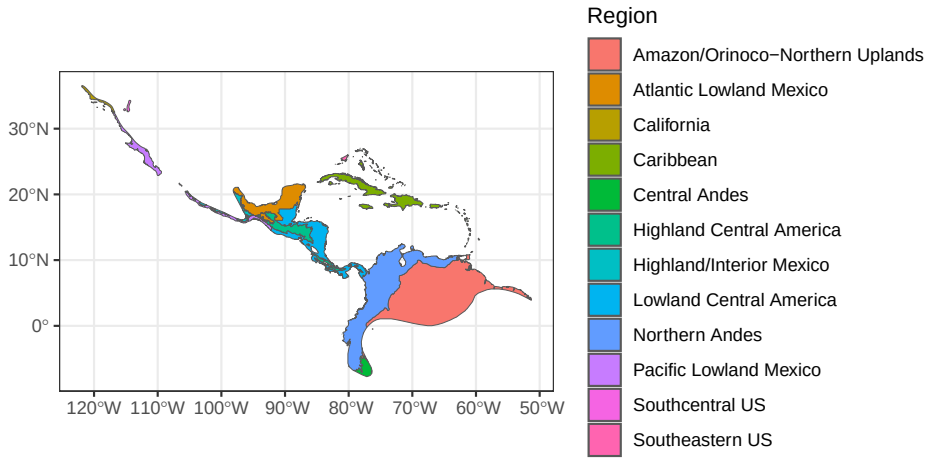
4.1 Subsetting ecoregions

The ecoregion data is derived from [provide link] and defines biogeographic regions in the Western Hemisphere. We provide these as external data stored as a SpatVector. We will intersect the ecoregions with the nonbreeding range of the American Redstart to identify all the ecoregions for this region.

```
library(mignette)
library(terra)
library(tidyterra)
library(tidyverse)
```

```
ecoregions <- system.file("extdata", "ecoregions.Rds",
                          package = "mignette") %>%
  terra::vect()
amre_nonbreeding_range <- system.file("extdata", "amre_nonbreeding_range.Rds",
                                     package = "mignette") %>%
  terra::vect()

amre_nonbreeding_ecoregions <- terra::intersect(amre_nonbreeding_range, ecoregions)

ggplot() +
  tidyterra::geom_spatvector(data = amre_nonbreeding_ecoregions,
                  aes(fill = Region)) +
```

```
  theme_bw()
```



4.2 Select sampled regions

For the purpose of the migratory network, we are only interested in nonbreeding nodes for which we have sampled data. To identify these nodes, we will intersect our sampled nonbreeding individuals locations with the ecoregion data. Additionally, since sampling locations may fall just outside an ecoregion boundary (for example, imprecise coordinates along a coast) we will identify the ecoregions by selecting the *nearest* ecoregion to a point.

```
nonbreeding_coords <- terra::vect(mignette::amre_nonbreeding_data,
                                  geom=c("Lon", "Lat"),
                                  crs = "EPSG:4326")
nonbreeding_coords_nearest <- terra::nearest(nonbreeding_coords,
                                             ecoregions,
                                             centroids = FALSE)
# Add Region ID from original file for easier interpretation
nonbreeding_coords_nearest$Region <- ecoregions$Region[nonbreeding_coords_nearest$to_id
```

```
table(nonbreeding_coords_nearest$Region)
```

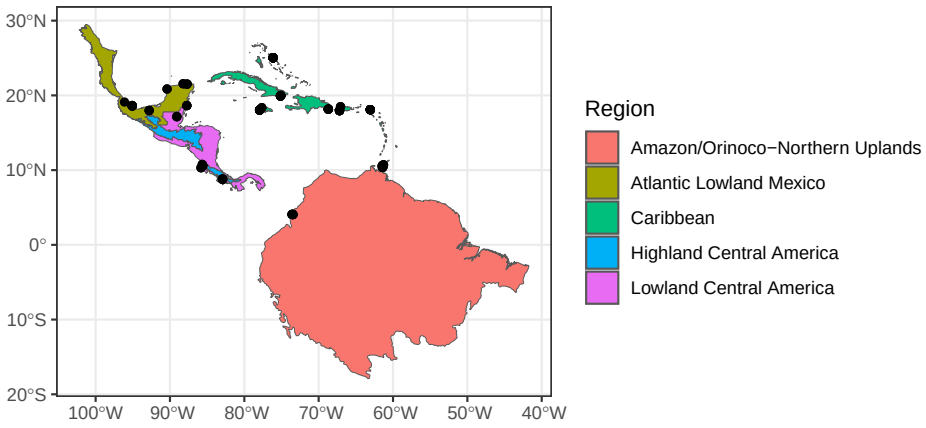
Var1	Freq
Amazon/Orinoco-Northern Uplands	21
Atlantic Lowland Mexico	22
Caribbean	69
Highland Central America	5
Highland/Interior Mexico	1
Lowland Central America	14
Northern Andes	3

For the migratory network, we also want sufficient sampling of a node for the model to be able to reach convergence. In the example above, we see "Highland/Interior Mexico" only has 1 sample and the "Northern Andes" only have 3 samples. Below, we use a cut off of a minimum of 4 samples to remove those ecoregions

```
cutoff <- 4

ecoregions_keep <- data.frame(table(nonbreeding_coords_nearest$Region)) %>%
  filter(!.[2] < cutoff) %>%
  pull(1) %>%
  as.character()

nonbreeding_coords_nearest_subset <- nonbreeding_coords_nearest[nonbreeding_coords_nearest$Regior
nonbreeding_regions_subset <- ecoregions[nonbreeding_coords_nearest_subset]
```

This leaves us with a final set of our nonbreeding nodes and the samples associated with them:

```
ggplot() +
  geom_spatvector(data = nonbreeding_regions_subset, aes(fill = Region)) +
  geom_spatvector(data = nonbreeding_coords_nearest_subset) +
  theme_bw()
```



Now on to getting the node assignment matrix.

Chapter 5

Node assignment

In `mignette`, node assignment data is one of the two main inputs into the migratory network model. Regardless of the methods used to collect these data, as input into the `mignette` migratory network model it needs to be in a specific format:

Breeding	CAR	AONU	ALM	HCA	LCA
BR	2	0	0	0	0
MP	0	9	0	0	0
NT	54	0	3	0	0
ST	12	12	0	4	1
WB	1	0	19	1	13

The first column specifies the *inferred* nodes in assignment. In the case of the genetic assignment data shown above, the breeding nodes are inferred from population assignment methods with individuals sampled from the nonbreeding range. The other column names refer to the nonbreeding node names, and the values are the number of individuals from a given nonbreeding node assigned to the respective breeding node specified by the row.

For the American Redstart example, we can get the assignment data in the format shown above based on population assignment data provided in `mignette` as well as the ecoregion information from the previous section.

The `amre_nonbreeding_range` data, from which we extracted the latitude and longitude coordinates of sampling locations to identify ecoregions in the previous section, also provides the breeding node assignment data from DeSaix et al. (2023). Creating the `nonbreeding_coords_nearest` object retained the same row order of data as in the original `amre_nonbreeding_range`, thus we can combine these data in the following way. **Note:** We are manually combining these data without a *join* by sample name or another identifier so it is always good to triple check that indeed the data are in the correct order when

not using a join.

```
amre_assign_manual <- tibble("Breeding" = amre_nonbreeding_data$Breeding_assignment,
                             "Nonbreeding" = nonbreeding_coords_nearest$Region) %>%
  filter(Nonbreeding %in% ecoregions_keep) %>% # remove those nonbreeding nodes with <
  group_by(Breeding, Nonbreeding) %>%
  summarize(N = n(),
            .groups = "drop") %>%
  pivot_wider(names_from = Nonbreeding, values_from = N) %>%
  rename("AONU" = "Amazon/Orinoco-Northern Uplands",
         "ALM" = "Atlantic Lowland Mexico",
         "CAR" = "Caribbean",
         "HCA" = "Highland Central America",
         "LCA" = "Lowland Central America") %>%
  mutate(across(where(is.numeric), ~ifelse(is.na(.), 0, .))) # replace NAs with 0s
```

```
amre_assign_manual
```

Breeding	CAR	AONU	ALM	HCA	LCA
BR	2	0	0	0	0
MP	0	9	0	0	0
NT	54	0	3	0	0
ST	12	12	0	4	1
WB	1	0	19	1	13

We can see that we have effectively mapped the breeding population assignment meta data to the correct ecoregion based on our individual's sampling location. Great!

Now on to getting the relative abundance.

Chapter 6

Relative Abundance

Here, we calculate the relative abundance for the different nodes of the migratory network. The input data for this step are:

- Abundance data (Raster)
- Population ranges (Raster or Vector)

Typically, the geographic range of a population will be stored as a vector (e.g., polygon), but in the case of a genoscape, the population ranges can be specified by a raster with q-values. For abundance data, we will use the eBird Status and Trends data which are obtained with the ebirdst R package, but any abundance data in a raster object can be used. **Note:** We do not provide the `ebirdst` data within `mignette` - these data are publicly available and licensed by the Cornell Lab of Ornithology and can be obtained by following the appropriate directions in the above links.

6.1 Get relative abundance for a genoscape (raster)

In the case of a *genoscape* raster with q-values, we will convert the q-values to probabilities of a pixel belonging to that population which is used to weight the abundance values. This is done with a single function `get_genoscape_abunds()`, and outputs a two-column matrix with the population names and the relative abundance of each population. Below, we provide an example with the genoscape as a SpatRaster (`genoscape`) and the ebirdst abundance data as a SpatRaster (`abunds`; obtained from the links above).

```
genoscape_abunds <- get_genoscape_abunds(genoscape = amre_genoscape, abunds = abunds)
```

6.2 Get relative abundance for a vector

In the case of vector file delineating populations, we sum abundance across the range of each population. This is done with a single function `get_vector_abunds()`, and outputs a two-column matrix with the population names and the relative abundance of each population. The example below provides the nonbreeding ecoregions file as a SpatVector (`winter_regions`) and the ebirdst abundance data as a SpatRaster (`abunds`)

```
winter_abunds <- get_vector_abunds(populations = nonbreeding_regions_subset, abunds = a
```

6.3 Combine the abundance data

As you'll see in the next section, these data are one of the primary inputs into the migratory network model and we combine the relative abundance data sets across the breeding and nonbreeding nodes with the following code. The final product is what we provide in this package

```
amre_abundance <- rbind(genoscape_abunds, winter_abunds) %>%
  as_tibble() %>%
  mutate(Relative_abundance = round(as.numeric(Relative_abundance)))
amre_abundance
```

Population	Relative_abundance
BR	2403
ST	9419
MP	19011
NT	72147
WB	26080
HCA	326
AONU	1139
LCA	2802
ALM	3169
CAR	7987

Now on to creating the migratory network model.

Chapter 7

Migratory Network Model

7.1 Input data

The input data we need for the migratory network model are:

- Relative abundance matrix for each node
- Assignment matrix of individuals among nodes

We demonstrate in the previous relative abundance section how to obtain the relative abundance data needed for the model. The relative abundance data need to be in the following format, with population IDs (same names as in the *assignment* file) in the first column and relative abundance values in the second column. Column names can follow any naming convention when inputting these data into `mignette`.

Population	Relative_abundance
BR	2403
ST	9419
MP	19011
NT	72147
WB	26080
HCA	326
AONU	1139
LCA	2802
ALM	3169
CAR	7987

For the assignment matrix, the input data needs to correspond to the following format:

Breeding	CAR	AONU	ALM	HCA	LCA
BR	2	0	0	0	0
MP	0	9	0	0	0
NT	54	0	3	0	0
ST	12	12	0	4	1
WB	1	0	19	1	13

where the first column specifies breeding population IDs while subsequent columns are the nonbreeding populations. The values are the number of individuals that connect each node. Depending on the type of assignment data the `mignette` user has, individual data points may need to be associated with a specific population. For these types of basic spatial analyses, `mignette` users can check out the `sf` package, specifically the `st_intersection()` function, or whatever geospatial tools they prefer to create these data. Once you have connectivity data in the above format, you can continue with the following sections.

For the following functions, we specify the order of the populations we are using for the model. Here, we are just ordering populations geographically by longitude to facilitate straightforward interpretation of the output.

```
bnode_names <- c("WB", "BR", "NT", "ST", "MP")
wnode_names <- c("ALM", "LCA", "HCA", "CAR", "AONU")
```

7.2 Migratory network model (JAGS)

7.2.1 Preparing input data

The following code provides the necessary data to run the JAGS model. To create the migratory network, the user first creates a text file specifying the JAGS model to be used, providing the name of the file to be saved (`base_filename`) and the type of model type (`model_type`). Currently `mignette` supports two model types based on the type of data used to determine assignment of individuals: `1` indicates that only genetic data were used for assignment, and `2` indicates that there's assignment data from both genetic and geolocator data. Here, the example only uses genetic data. `get_jags_model()` saves a `.txt` file with the `base_filename` and stores that name as a variable for use in JAGS. We also specify the desired order of the breeding populations (`bnode_names`) and the nonbreeding populations (`wnode_names`). Finally, we use these as input into the function `get_jags_data()` to prepare the data appropriately for the model.

```
base_filename <- mignette::get_jags_model(base_filename = "amre.genetic.model", model_t
jags_data <- mignette::get_jags_data(abundance = mignette::amre_abundance,
                                     assignment = mignette::amre_assign,
                                     bnode_names = bnode_names,
                                     wnode_names = wnode_names)
```

7.2.2 Running the model

Now the user can use the output of `jags_data` into JAGS to run the actual model:

```
parameters <- c("conn_g")
ni <- 500000
nt <- 4
nb <- 100000
nc <- 2
jags_out <- autojags(jags_data, inits=NULL, parameters, paste0(base_filename,".txt"),
                     n.chains = nc, n.thin =nt, iter.increment=ni,
                     max.iter = ni*50+nb, n.burnin = nb,
                     n.adapt= NULL, parallel=TRUE)
amre_conn <- jags_out$mean$conn_g
colnames(amre_conn) <- wnode_names
rownames(amre_conn) <- bnode_names
amre_conn
```

	ALM	LCA	HCA	CAR	AONU
WB	0.14212	0.13901	0.01346	0.00507	0.00002
BR	0.00010	0.00021	0.00008	0.02082	0.00004
NT	0.05680	0.00041	0.00013	0.38276	0.00006
ST	0.00003	0.01345	0.03618	0.03902	0.06453
MP	0.00013	0.00036	0.00022	0.00002	0.08496

The connectivity between the nodes is provided by the `conn_g` parameter of the model, which is accessed from the above code. The rows are the breeding nodes (the *inferred* population from assignment) and the columns are the nonbreeding nodes (the *sampled* population).

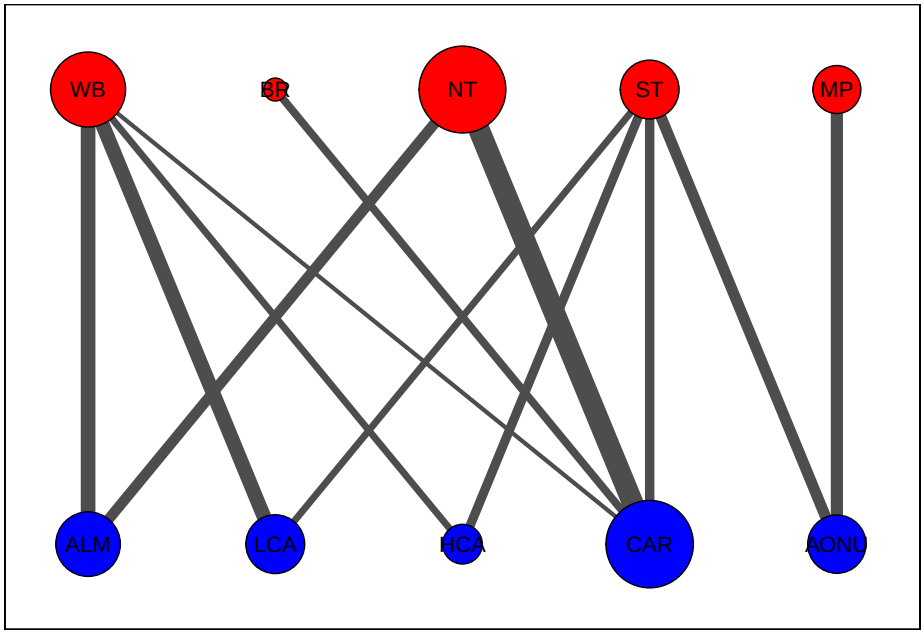
Voila. A migratory network model is completed. Now, onto visualization

Chapter 8

Network visualization

We provide functions for basic visualization of the network (`net_create()` and `net_draw()`). A threshold can be set to remove very weak connectivity for better visualizing the network as shown below.

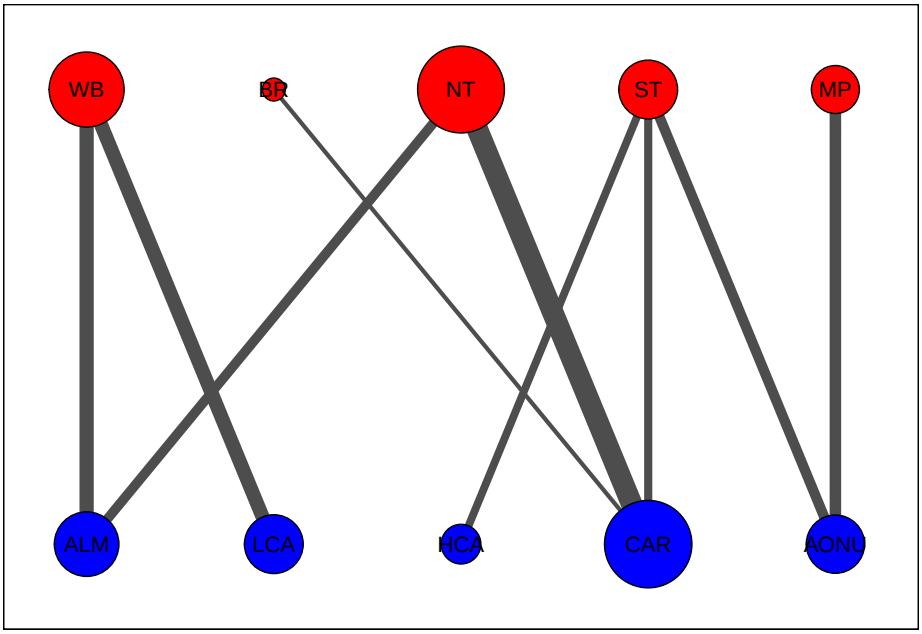
```
bnode_names <- c("WB", "BR", "NT", "ST", "MP")
wnode_names <- c("ALM", "LCA", "HCA", "CAR", "AONU")
net <- mignette::net_create(mignette::amre_conn,
                            node.names = list(bnode_names, wnode_names))
#set the display size range for nodes (min and max), default 1-10
net$display_par$node_size_scale<-c(5,20)
#set the display size range for edges (min and max), default 1-10
net$display_par$edge_size_scale<-c(1,5)
plot(mignette::net_draw(net))
```



In this visualization, node size corresponds to the amount of connectivity with that population and edge size corresponds to the amount of connectivity between the populations. By default, breeding populations are in the top row (red) and nonbreeding populations are in the bottom row (blue).

We also can change the `connected_tol` threshold in the `net_create` function to not output extremely weak connectivity. For example, the plot below removes connectivity less than 0.02:

```
net <- mignette::net_create(mignette::amre_conn,
                            node.names = list(bnode_names, wnode_names),
                            connected_tol = 0.02) # threshold
net$display_par$node_size_scale<-c(5,20)
net$display_par$edge_size_scale<-c(1,5)
plot(mignette::net_draw(net))
```



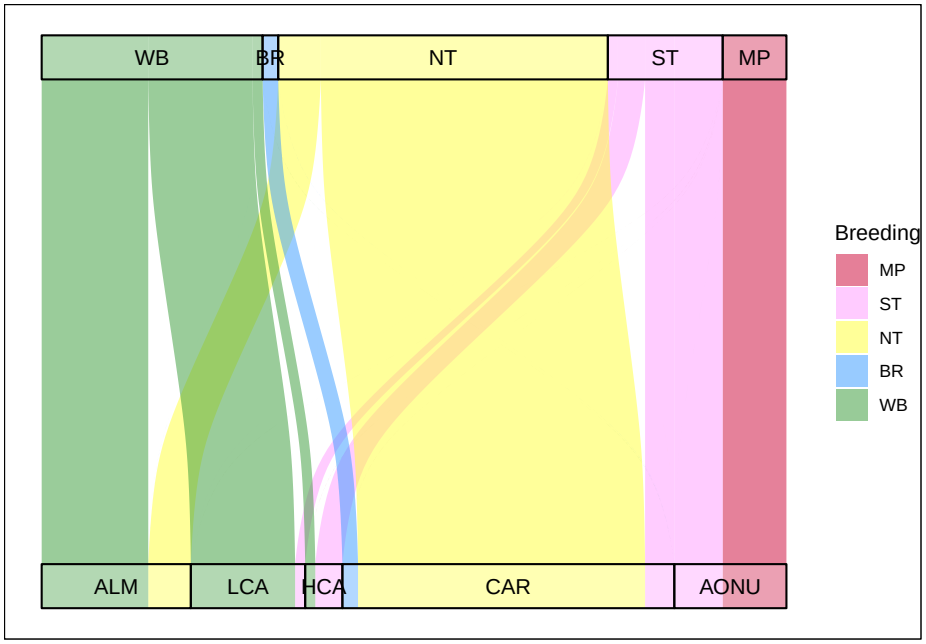
This sums up the basics of creating and visualizing a migratory network. We encourage users to explore and build upon the visualization tools we provide (e.g. overlay the migratory networks on geographic ranges) - the options are endless, enjoy!

8.1 Other visualization options

Ok, we'll toss out another way to visualize these networks using the `ggalluvial` package because it's really fun.

```
library(tidyverse)
library(mignette)
library(ggalluvial)

bnode_names <- c("WB", "BR", "NT", "ST", "MP")
wnode_names <- c("ALM", "LCA", "HCA", "CAR", "AONU")

cluster_colors <-  c(
  `ST` = "#FF99FF", # pink/Southern Temperate
  `BR` = "#3399FF", # blue/Basin Rockies
  `NT` = "#FFFF33", # yellow/Northern Temperate
  `WB` = "#339933", # green/Western Boreal
  `MP` = "#CC0033" # red/Maritime Provinces
)
```

```
amre_conn_df <- mignette::amre_conn %>%
  as_tibble(rownames = "Breeding") %>%
  pivot_longer(cols = ALM:AONU, names_to = "Nonbreeding", values_to = "Connectivity") %
  mutate(Connectivity = ifelse(Connectivity < 0.01, 0, Connectivity)) %>%
  mutate(Breeding = factor(Breeding, levels = rev(bnode_names)),
         Nonbreeding = factor(Nonbreeding, levels = rev(wnode_names)))

p.alluvial <- ggplot(amre_conn_df,
       aes(y = Connectivity, axis1 = Nonbreeding, axis2 = Breeding)) +
  geom_alluvium(aes(fill = Breeding), width = 1/12) +
  scale_fill_manual(values = cluster_colors) +
  geom_stratum(alpha = 0.25, width = 1/12) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Nonbreeding", "Breeding"),
                   expand = c(0.05, 0.05)) +
  coord_flip() +
  theme_void() +
  theme(plot.background = element_rect(fill = "white"))

p.alluvial
```



Bibliography

Matthew G DeSaix, Lesley P Bulluck, Andrew J Eckert, Catherine B Viverette, Than J Boves, Jessica A Reese, Christopher M Tonra, and Rodney J Dyer. Population assignment reveals low migratory connectivity in a weakly structured songbird. *Molecular Ecology*, 28(9):2122–2135, 2019.

Kristen C Ruegg, Eric C Anderson, Kristina L Paxton, Vanessa Apkenas, Sirena Lao, Rodney B Siegel, David F DeSante, Frank Moore, and Thomas B Smith. Mapping migration in a songbird using high-resolution genetic markers. *Molecular Ecology*, 23(23):5726–5739, 2014.

Kristen C Ruegg, Ryan J Harrigan, James F Saracco, Thomas B Smith, and Caz M Taylor. A genoscape-network model for conservation prioritization in a migratory bird. *Conservation Biology*, 34(6):1482–1491, 2020.

Kristen C Ruegg, Michaela Brinkmeyer, Christen M Bossu, Rachael A Bay, Eric C Anderson, Clint W Boal, Russell D Dawson, Amber Eschenbauch, Christopher JW McClure, Karl E Miller, et al. The american kestrel (falco sparverius) genoscape: Implications for monitoring, management, and subspecies boundaries. *The Auk*, 138(2):ukaa051, 2021.

Caz M Taylor and D Ryan Norris. Population dynamics in migratory networks. *Theoretical Ecology*, 3:65–73, 2010.