Creating Migratory Networks in R

Matt DeSaix

2023-05-17

Contents

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 [Ruegg et al., 2020, Taylor and Norris [2010]]. The core function of `mignette` is to model the connectivity of populations across the annual cycle. **To model these migratory networks in `mignette`, users need to provide:**

1. Assignment (or connectivity) data for the number of individuals connecting migratory populations from different 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, and we use genetic markers to delineate breeding populations (i.e., nodes in the breeding 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 out of 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 a brief demonstration of creating a migratory network from assignment and relative abundance data, 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:

```r
# 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:

```r
library(mignette)
library(sf)
library(terra)
library(tidyverse)
library(ebirdst)
library(rjags)
library(jagsUI)
library(ggnewscale) # for network visualization
```

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
NT	54	0	3
ST	12	12	0
WB	1	0	20

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
ST	9874.3106
NT	75628.5485
WB	27340.7038
AONU	942.6022
ALM	1667.3713
CAR	4850.6727

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", "NT", "ST")
wnode_names <- c("ALM", "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.

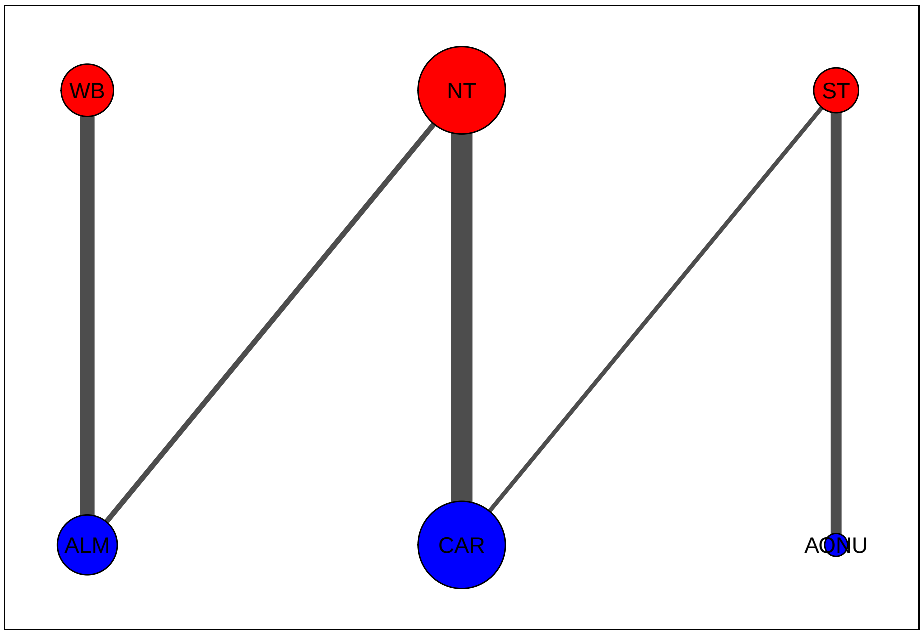
```
base_filename <- mignette::get_jags_model(base_filename = "amre.genetic.model", model_
jags_data <- mignette::get_jags_data(abundance = mignette::amre_abundance,
                                     assignment = mignette::amre_assign,
                                     bnode_names = bnode_names,
                                     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 can be set to remove very weak connectivity for better visualizing the network as shown below.

```
# set threshold for not visualizing minimal connectivity
amre_conn_test <- mignette::amre_conn
amre_conn_test[amre_conn_test < 0.01] <- 0
net <- mignette::net_create(amre_conn_test,
                            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/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:

```r
# 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:

```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 `comyel_breeding_data` data set provides admixture results (Q-values) of five genotype clusters for Common Yellowthroat (cite a coye paper) and metadata for the sampled individuals.

```r
Q_matrix <- Mignette::comyel_breeding_data %>%
  dplyr::select(CA, Midwest, NewEngland, West, Southwest) %>%
  as.matrix()
long_lat_matrix <- Mignette::comyel_breeding_data %>%
  dplyr::select(Long, Lat) %>%
  as.matrix()
cluster_colors <-  c(
  CA = "#CC0000",
  Midwest = "#3399FF",
  NewEngland = "#9933CC",
  West = "#009933",
  Southwest = "#FF6600")
```

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

```r
# remotes::install_github("eriqande/TESS3_encho_sen")
genoscape_brick <- tess3r::tess3Q_map_rasters(
  x = Q_matrix,
  coord = long_lat_matrix,
  map.polygon = breed_smooth,
  window = terra::ext(breed_smooth)[1:4],
  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(genoscape_brick) <- colnames(Q_matrix)

out.files <- paste0("./comyel/genoscape/comyel_genoscape_cluster_", names(genoscape_brick), ".tif
terra::writeRaster(terra::rast(genoscape_brick), filename = out.files)
```

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 section if you are ready to do that with the genoscape. Or if you still need to create the wintering nodes, check out the wintering nodes section. 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

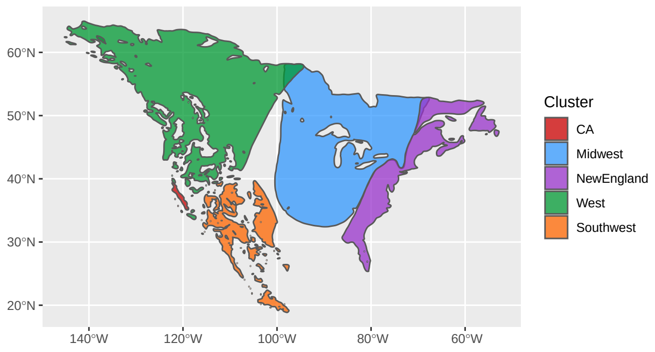
Using the genoscape rasters we will convert them to polygons, using the handy `scape_to_shape()` function. The `prob_threshold` parameter specifies the value to determine if a raster cell is included in the polygon for that genoscape. This value should be customized for different species to check for 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.

```
genoscape_files <- list.files("./comyel/genoscape",
                                    full.names = T,
                                    pattern = "*.tif")

genoscape_raster_stack <- terra::rast(genoscape_files)
genoscape_polygon_sf <- mignette::scape_to_shape(x = genoscape_raster_stack, prob_threshold = 0.5
```

Check out the polygons

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



Check which polygons are overlapping. Each row of the output provides a pair of overlapping polygons (if there are any).

```
mignette::check_genoscape_overlap(genoscape_polygon_sf)
```

```
##      [,1]      [,2]
## [1,] "CA"      "West"
## [2,] "Midwest" "NewEngland"
## [3,] "Midwest" "West"
```

Now on to creating the wintering nodes

Chapter 4

Wintering 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 nonbreeding of nodes interest, then continue to the relative abundance section.

4.1 Subsetting winter ecoregions

The ecoregion data is provided by [provide link]. The ecoregions are provided in this this package as `ecoregions_simple`. We will intersect the ecoregions with the wintering range of the Common Yellowthroat to identify all the ecoregions that overlap with the wintering range.

```
comyel_winter <- comyel_range_smooth %>%
  dplyr::filter(season == "nonbreeding") %>%
  sf::st_transform(crs = 4326) %>%
  sf::st_intersection(mignette::ecoregions_simple) %>%
  dplyr::select(Region)
```

4.2 Snap points

Sometimes individuals are not quite within the wintering nodes. Here, we will make sure all sampled individuals get assigned to the nearest ecoregion with the `mignette` function `snap_points()`.

```
winter.coords <- mignette::comyel_wintering_data %>%
  st_as_sf(coords=c("Long","Lat")) %>%
  st_cast("MULTIPOINT") %>%
  st_set_crs(4326)
```

```
new.winter.coords <- mignette::snap_points(winter.coords, comyel_winter, 150000)
```

4.3 Finalize wintering nodes

Now that points have been *snapped* to the appropriate ecoregions, we can further subset all of the ecoregions in which we have actually sampled individuals from. If we haven't sampled individuals from a region, we can't use that region as a node in the migratory connectivity network!

```
winter_intersect <- st_intersects(comyel_winter, new.winter.coords, sparse = T)
poly_id <- which(unlist(lapply(winter_intersect, function(x) length(x) > 0)))
comyel_winter_ecoregions <- comyel_winter[poly_id,]
```

Now on to getting the relative abundance.

Chapter 5

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, but any abundance data in a raster object can be used.

5.1 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.

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

5.2 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_genoscape_abunds()`, and outputs a two-column matrix with the population names and the relative abundance of each population.

```
winter_abunds <- get_vector_abunds(populations = winter_regions, abunds = abunds)
```

Chapter 6

Migratory Network Model

6.1 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
ST	9874.3106
NT	75628.5485
WB	27340.7038
AONU	942.6022
ALM	1667.3713
CAR	4850.6727

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

Breeding	CAR	AONU	ALM
NT	54	0	3
ST	12	12	0
WB	1	0	20

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", "NT", "ST")
wnode_names <- c("ALM", "CAR", "AONU")
```

6.2 Migratory network model (JAGS)

6.2.1 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_
jags_data <- mignette::get_jags_data(abundance = mignette::amre_abundance,
                                     assignment = mignette::amre_assign,
                                     bnode_names = bnode_names,
                                     wnode_names = wnode_names)
```

6.2.2 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
amre_conn
```

0.2219310	0.0087849	0.0000857
0.0557115	0.5343810	0.0001573
0.0000205	0.0538838	0.1250444

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 and the columns are the nonbreeding nodes, which we can make more apparent:

```
colnames(amre_conn) <- wnode_names
rownames(amre_conn) <- bnode_names
amre_conn
```

```
##             ALM         CAR         AONU
## WB 2.219310e-01 0.008784863 8.568762e-05
## NT 5.571149e-02 0.534380974 1.573024e-04
## ST 2.046623e-05 0.053883752 1.250444e-01
```

	ALM	CAR	AONU
WB	0.2219310	0.0087849	0.0000857
NT	0.0557115	0.5343810	0.0001573
ST	0.0000205	0.0538838	0.1250444

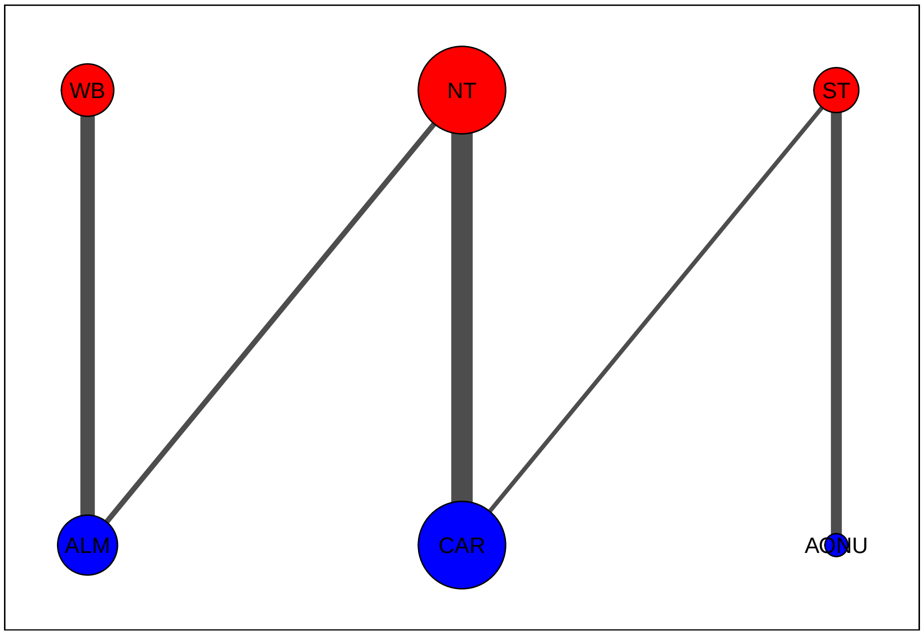
Voila. A migratory network model is completed. Now, onto visualization

Chapter 7

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.

```r
bnode_names <- c("WB", "NT", "ST")
wnode_names <- c("ALM", "CAR", "AONU")
# set threshold for not visualizing minimal connectivity
amre_conn_test <- mignette::amre_conn
amre_conn_test[amre_conn_test < 0.01] <- 0
net <- mignette::net_create(amre_conn_test,
                             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/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!

Bibliography

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.