Creating Migratory Networks in R: mignette

2024-09-10

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 using migratory networks [Ruegg et al., 2020, Taylor and Norris, 2010]. Spatial migratory networks are graph-based models that incorporate migratory connectivity and abundance data to define the degree of connectivity between stages of the annual cycle, defined as breeding and nonbreeding nodes. These models are useful for a wide variety of applications because they can accommodate, and combine, different types of common tracking data used to infer individual and population movement across the annual cycle. We designed `mignette` to facilitate the creation and visualization of migratory networks for users with a wide-range of data sources and study species.

1.1 Tutorial structure

For an introductory demonstration of creating a migratory network with `mignette`, see the quick start example. This demonstrates the core functionality of `mignette` using data from a recent study of the American Redstart (*Setophaga ruticilla*), a widespread Nearctic-Neotropical migratory songbird [DeSaix et al., 2023].

Subsequent chapters of the vignette provide more detailed instructions and are broken up into three main steps:

1. Define breeding and nonbreeding nodes
2. Assemble abundance and migratory connectivity data
3. Modeling the migratory network

1.2 Installation

While `mignette` is an R package, it uses JAGS (Just Another Gibbs Sampler) to actually implement the migratory network model. JAGS is a specific software for conducting analysis of Bayesian hierarchical models using Markov Chain Monte Carlo simulation. **JAGS needs to be installed for `mignette` to function** and can be downloaded here.

Once you have installed JAGS, you can install `mignette` from GitHub with:

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

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
```

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:

- Abundance
- Migratory connectivity

We provide an example of these data in `mignette`, with migratory connectivity data from 5 breeding nodes (WB = Western Boreal, NT = Northern Temperate, ST = Southern Temperate, BR = Basin Rockies, MP = Maritime Provinces) and 5 nonbreeding nodes (ALM = Atlantic Lowland Mexico, CAR = Caribbean, AONU = Amazon/Orinoco-Northern Uplands, HCA = Highland Central America, LCA = Lowland Central America) for the American Redstart (*Setophaga ruticilla*). The migratory connectivity data 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

We also provide the abundance of these nodes:

```
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

Network model

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

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

For the American Redstart migratory network, we use `model = BR` which specifies that nonbreeding nodes are the "encounter" season and breeding nodes are the "recovery" season (i.e., inferred). This output saves the model as `amre.genetic.model_BR.txt`. Below we specify `parallel = TRUE` to run MCMC on multiple cores and use the remaining defaults. This step is computationally intensive and takes ~2 minutes to run on a 2023 MacBook Pro with an Apple M2 Pro chip.

```
network_model <- run_network_model(abundance = amre_abundance,
                                   nb2br_assign = amre_assign,
                                   brnode_names = brnode_names,
                                   nbnode_names = nbnode_names,
                                   model = "BR", base_filename = "amre.genetic",
                                   parallel = TRUE)
```

The first component of the output, `[["conn"]]`, is an R tibble object of the mean connectivity estimated between nodes (Table 1). These values are interpreted as

the proportion of the global population that migrate between the corresponding populations, as such all of the values in the network matrix sum to one. The second component, `[["jags_out"]]`, is the full output from `jagsUI::autojags()` provided as a list object, which contains important model information such as parameter estimates and credible intervals, model specifications, and goodness of fit. The final two components, `[["brnode_names"]]` and `[["nbnode_names"]]`, store the node names corresponding to the rows and columns, respectively, of the connectivity matrix.

```
network_model$conn
```

Breeding	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 second component is the full output from `*jagsUI* autojags()` and is accessed by `network_model$jags_out`. Here, the raw connectivity matrix (i.e., mean *conn_g* estimates) can be accessed:

```
amre_conn <- network_model$jags_out$mean$conn_g
amre_conn
```

0.14212	0.13901	0.01346	0.00507	0.00002
0.00010	0.00021	0.00008	0.02082	0.00004
0.05680	0.00041	0.00013	0.38276	0.00006
0.00003	0.01345	0.03618	0.03902	0.06453
0.00013	0.00036	0.00022	0.00002	0.08496

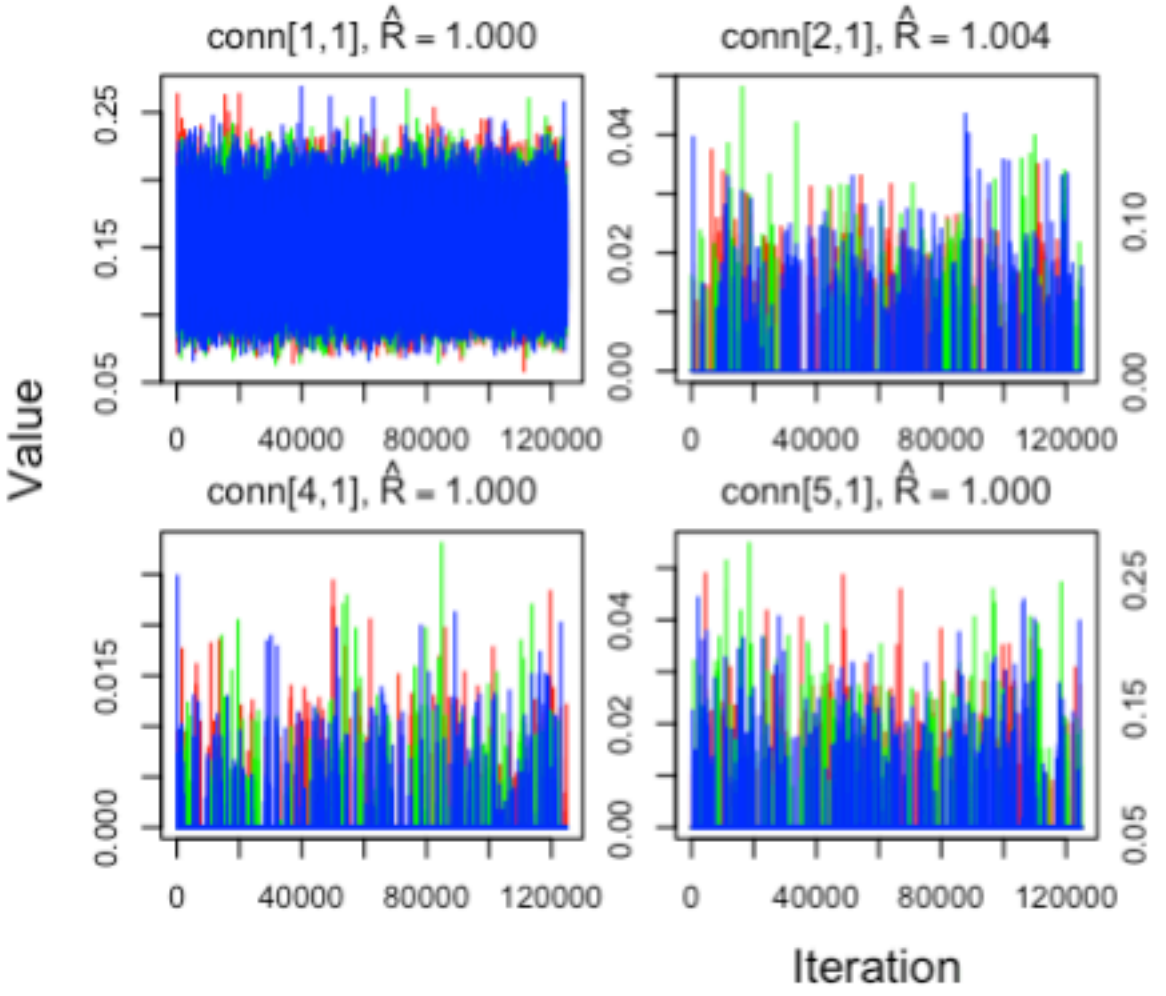
Check parameter convergence

`mignette` users should familiarize themselves with the `jagsUI autojags()` output in order to evaluate the network model appropriately. For example, all parameters of the model should converge ($Rhat < 1.1$) and this can be double-checked by counting the number of instances where there are *Rhat* values greater than or equal to 1.1 with:

```
sum(network_model$jags_out$Rhat$conn_g >= 1.1)
[1] 0
```

In this case, all of the *Rhat* values for the connectivity estimates were less than 1.1 and thus the result was 0 - all parameters converged! In the absence of convergence, the MCMC iterations may need to be increased. Additionally, convergence should be inspected visually to ensure that all MCMC chains have stabilized.

```
jagsUI::traceplot(network_model$jags_out, "conn", layout = c(2,3))
```

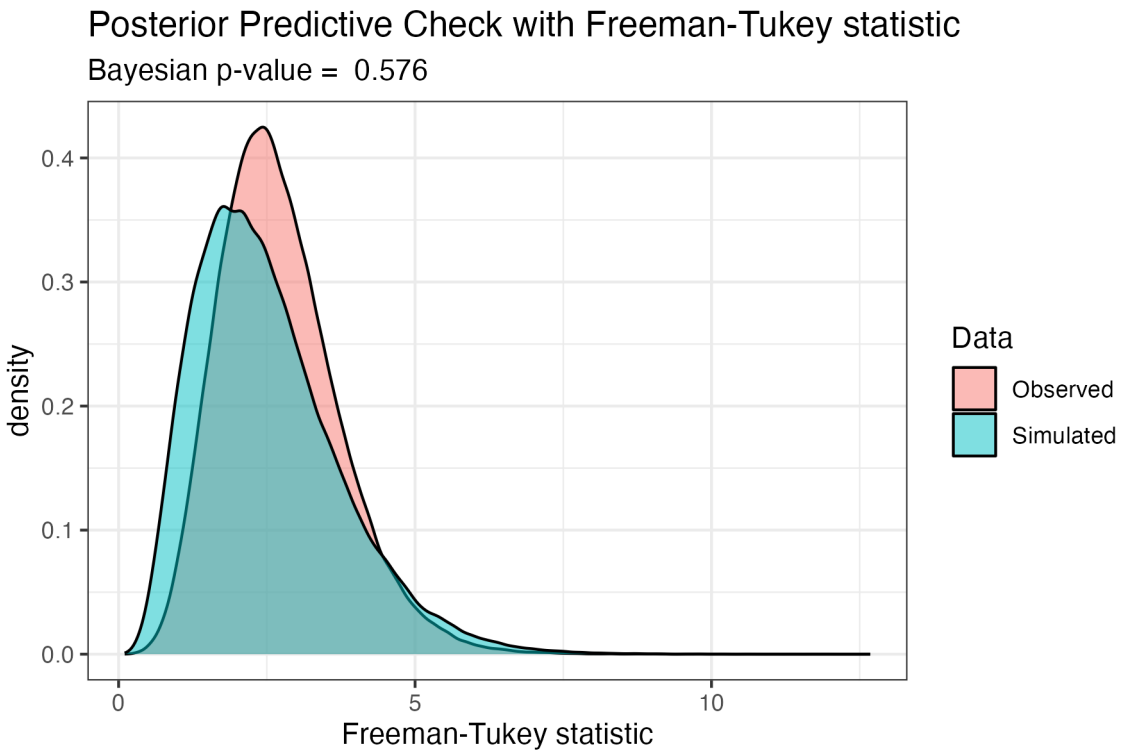


Check goodness of fit

In *mignette*, we implement a posterior predictive check using the Freeman-Tukey discrepancy statistic to compare the fit of the observed data to simulated data from the model, following the guidelines of Conn et al. (2018). In the network model output, the posterior predictive check parameters are: 1) FT.obs = the Freeman-Tukey statistic for observed data, and 2) FT.rep = the Freeman-Tukey statistic for simulated data. The function `get_FT_fit()` produces a density plot of the Freeman-Tukey statistics for qualitatively comparing observed and simulated data, as well as outputting the Bayesian p-value of the goodness of fit based on these distributions. Very small (<0.05) or large (>0.95) Bayesian p-values suggest a lack of fit of the model. For a full discussion of checking Bayesian models, see Conn et al. (2018).

We can then look at the posterior predictive check of the American Redstart mode using the code below, which shows largely overlapping distributions of the Freeman-Tukey discrepancy statistics for the observed and simulated data, with a corresponding Bayesian p-value of 0.58 - indicating sufficient goodness of fit.

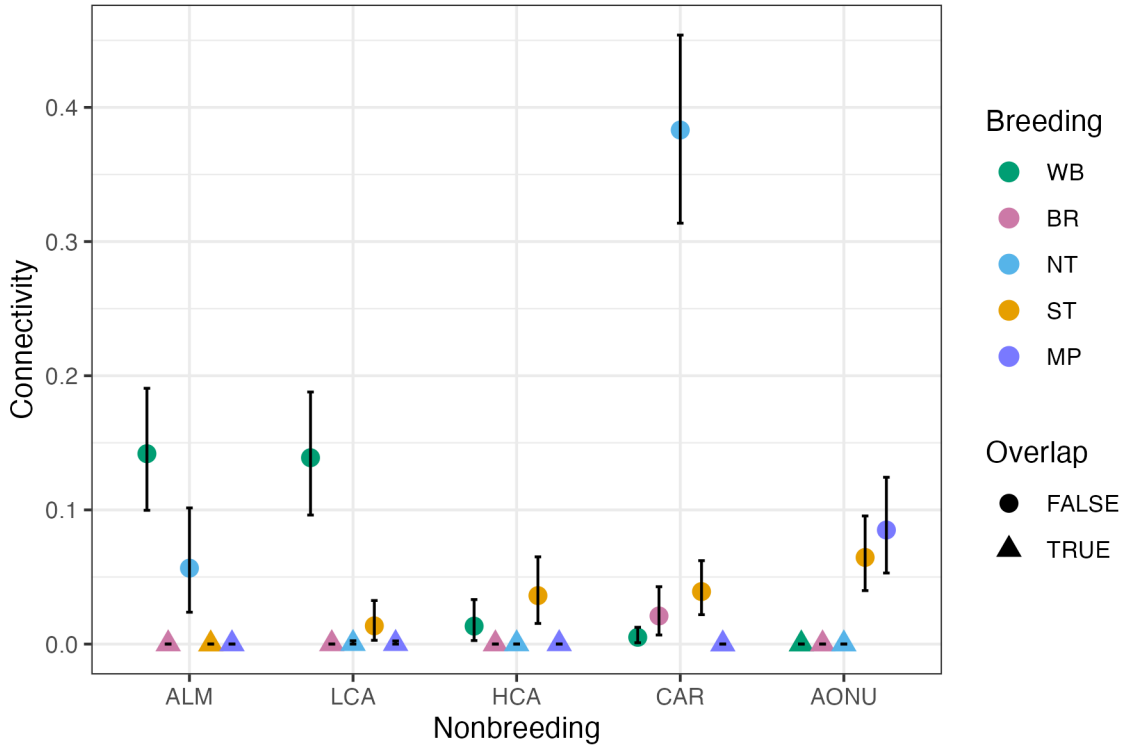
```
get_FT_fit(network_model)
```



Uncertainty in the connectivity estimates

The network model output provides credible intervals in addition to the mean of the connectivity estimates. Uncertainty in the network connectivity estimates is characterized by the credible intervals provided in the network model output for the conn_g parameter. The mignette function `plot_network_CI()` plots the mean and 95% credible intervals for network connectivity which allows users to assess the uncertainty in the estimates.

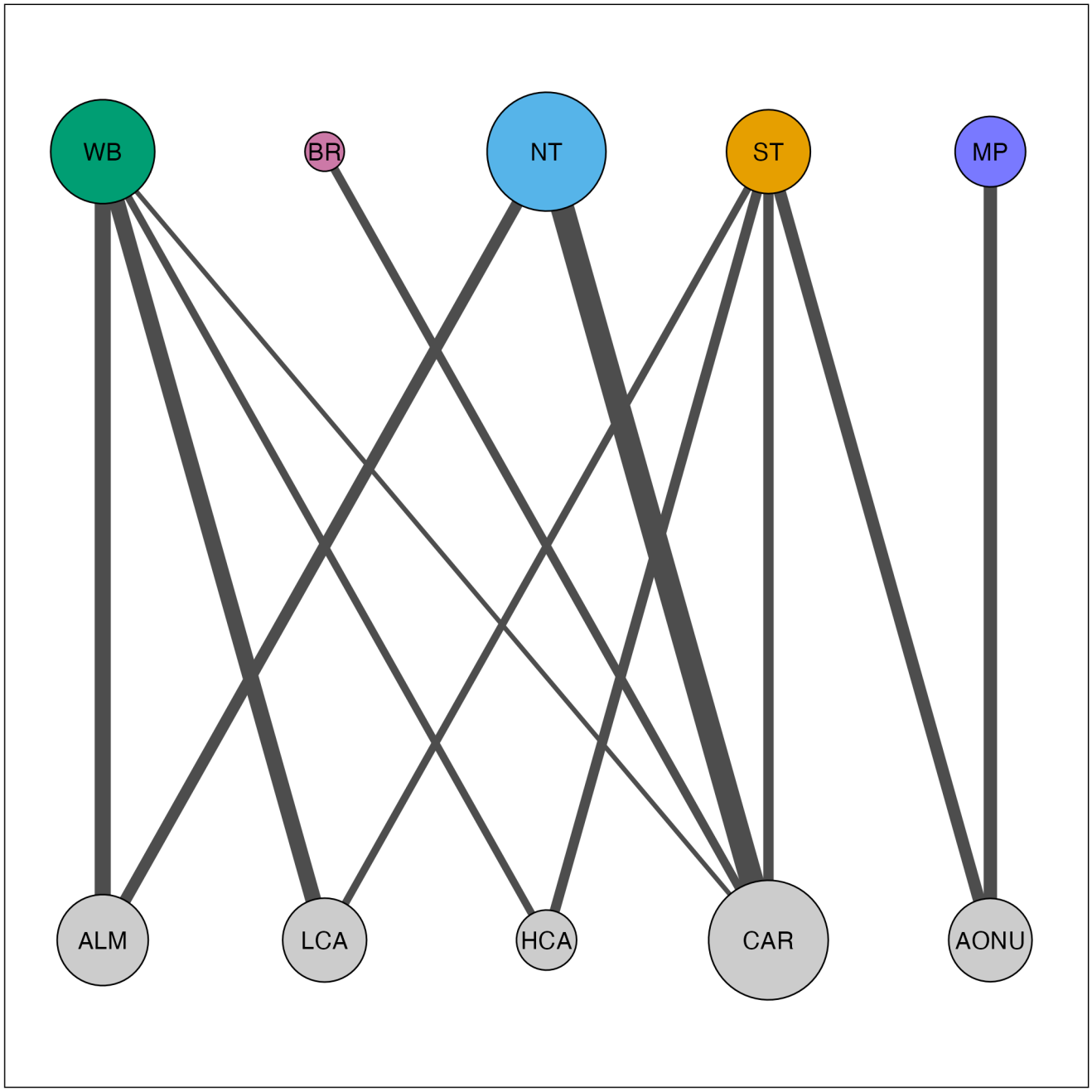
```
plot_network_CI(network_model = network_model, stage = "Breeding",
stage_colors = brnode_colors, overlap = TRUE)
```



Visualize network

The raw connectivity matrix is used to plot the network. We plot the migratory network with the provided `mignette` functions `net_create()` and `net_draw()`. We set `connected_tol = 0.01` which plots only the edges with connectivity values of greater than 0.01.

```
amre_net <- net_create(network_model = network_model,
                       margin = 0.05)
#set the display size range for nodes (min and max), default 1-10
amre_net$display_par$node_size_scale <- c(8,25)
#set the display size range for edges (min and max), default 1-10
amre_net$display_par$edge_size_scale <- c(1,5)
# change colors
amre_net$display_par$brnode_colors <- c("#009e73", "#cc79a7", "#56b4e9", "#e69f00", "#
amre_net$display_par$nbnode_colors <- "grey80"

net_draw(amre_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. Breeding populations are in the top row, for which we provided

custom colors, and nonbreeding populations are in the bottom row.

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

Define breeding and nonbreeding nodes

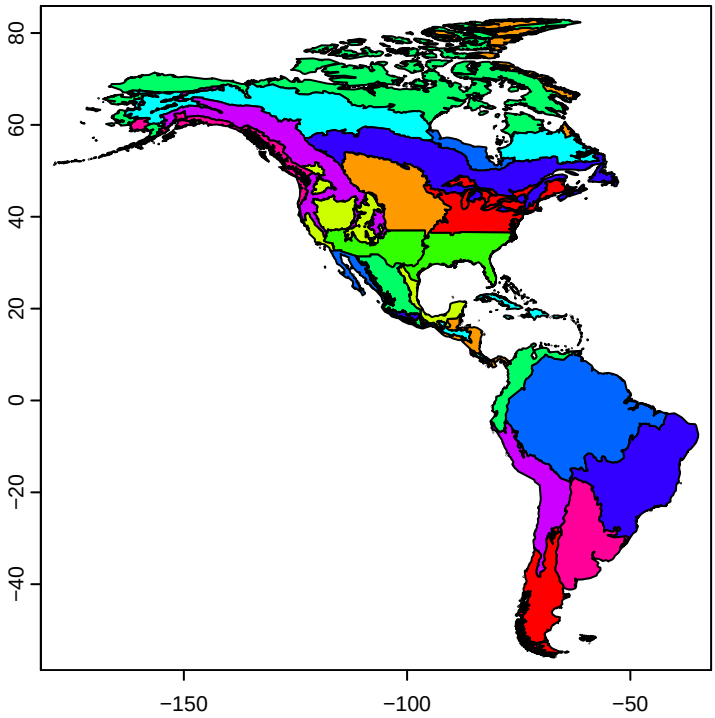
The first step of creating a migratory network is to spatially define the nodes (i.e. populations) that make up the sampled portions of the breeding and nonbreeding range. This can either be done with *vector* data for discrete spatial boundaries of nodes or *raster* data for a continuous surface probabilistic membership of geographic cells to a node (e.g., population genetic structure of a genoscape [Ruegg et al., 2020]).

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

3.1 Vector data

Vector data can define nodes by representing geographic boundaries such as political borders or biogeographic regions. In `mignette`, we provide data for *conservation regions* of the Western Hemisphere that integrate geopolitical boundaries and ecoregions.

```
# SpatVector from terra package
regions_file <- system.file("extdata", "conservation_regions.Rds", package = "mignette")
# read in file as SpatVector
conservation_regions <- terra::vect(regions_file)

terra::plot(conservation_regions, col=rainbow(10))
```



For defining a node, we are only interested in regions that we have data. It's straightforward with the spatial functions from `terra` to determine the sampled regions. Here, we find the conservation regions for which we have sampled American Redstart data from the nonbreeding range.

```
# create SpatVector points object from Redstart lat/lon sampling data
nonbreeding_coords <- terra::vect(mignette::amre_nonbreeding_data,
                                  geom=c("Lon", "Lat"),
                                  crs = "EPSG:4326")
# find spatial intersection of points with conservation regions
sampled_conservation_regions <- terra::intersect(nonbreeding_coords,
                                                 conservation_regions)
# count the number of sampled individuals associated with each conservation region
sampled_table <- table(sampled_conservation_regions$Region)
sampled_table
```

```
##
## Amazon/Orinoco-Northern Uplands         Atlantic Lowland Mexico
##                              20                              15
##                       Caribbean       Highland Central America
##                              51                               5
##       Highland/Interior Mexico        Lowland Central America
##                               1                              14
```

```
##                   Northern Andes
##                                3
```

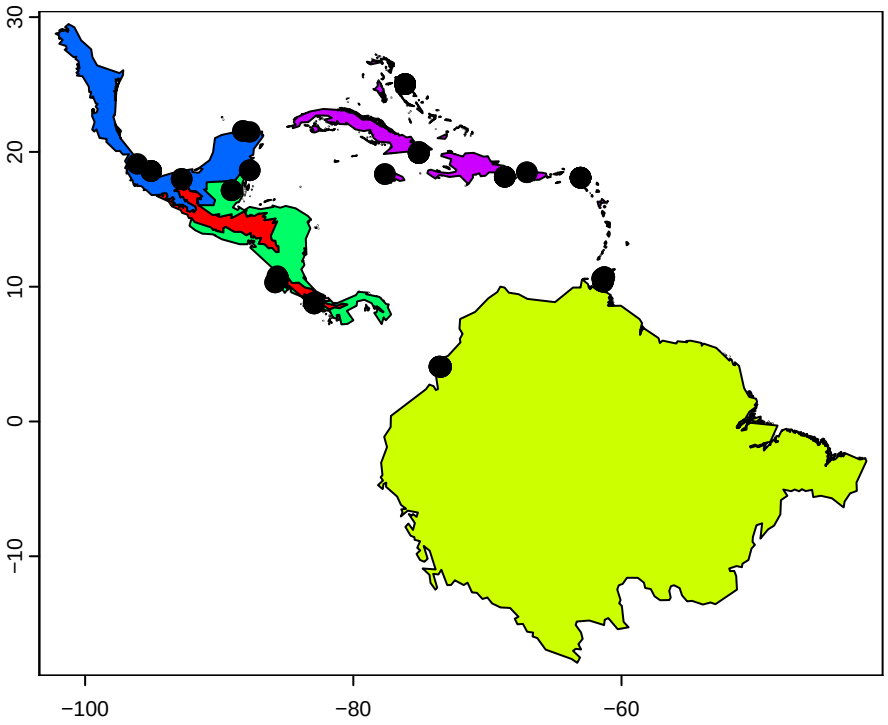
Downstream in the workflow, we need sufficiently sampling of individuals in (or inference to) a node. So, we'll only retain the conservation regions with more than 3 samples.

```
# retain conservation region names with more than 3 samples
retained_IDs <- names(sampled_table[sampled_table > 3])
retained_IDs
```

```
## [1] "Amazon/Orinoco-Northern Uplands" "Atlantic Lowland Mexico"
## [3] "Caribbean"                       "Highland Central America"
## [5] "Lowland Central America"
```

Thus, we've identified the 5 nonbreeding nodes for the American Redstart data: Amazon/Orinoco-Northern Uplands (AONU), Atlantic Lowland Mexico (ALM), Caribbean (CAR), Highland Central America (HCA), and Lowland Central America (LCA). We can plot these conservation regions along with the corresponding sampling points.

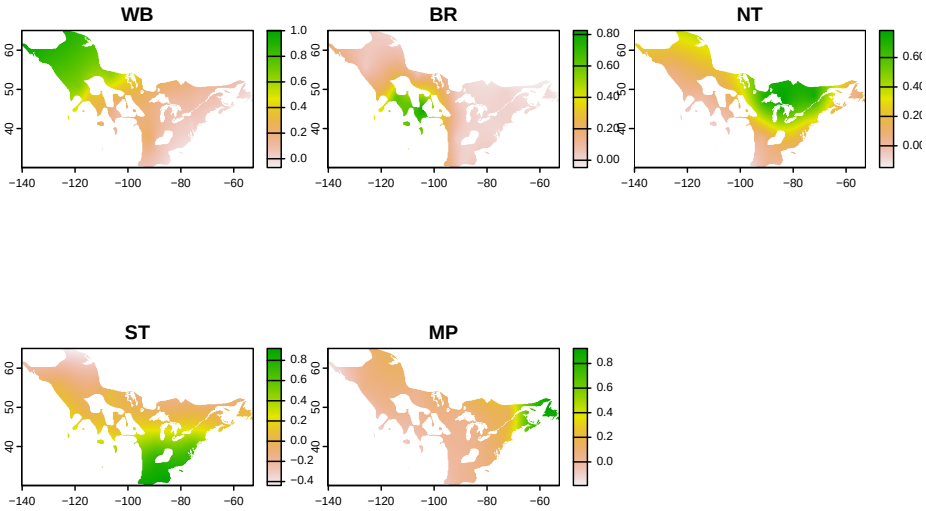
```
# subset conservation regions file by retained regions
conservation_regions_subset <- terra::subset(conservation_regions,
                                             conservation_regions$Region %in% retained_IDs)
# subset sampled points by retained regions
nonbreeding_coords_subset <- terra::subset(sampled_conservation_regions,
                                           sampled_conservation_regions$Region %in% retained_IDs)
# plot
terra::plot(conservation_regions_subset, col = rainbow(5))
terra::points(nonbreeding_coords_subset, cex = 1.5)
```



3.2 Raster data

Raster data is used when populations are defined by a continuous surface of probabilities of membership to a population. In the example here, we have population genetic data from the American Redstart that define breeding populations (i.e., American Redstart genoscape, [DeSaix et al., 2023]). We have defined these nodes as Western Boreal (WB), Basin Rockies (BR), Northern Temperate (NT), Southern Temperate (ST) and Maritime Provinces (MP). Raster data for nodes are used in `mignette` to get node abundance that is scaled by the raster probabilities of population membership, as shown in the next section.

```
amre_genoscape_file <- system.file("extdata", "amre_genoscape.Rds", package = "mignett
amre_genoscape <- terra::rast(amre_genoscape_file)
terra::plot(amre_genoscape)
```



While it is outside the scope of `mignette` to perform the requisite population genetics analyses for producing these rasters, users with genetic data and expertise with genetic clustering analyses may be interested in the Supplementary information that provide additional information and code.

Now on to Step 2) Abundance and migratory connectivity data

Chapter 4

Assemble abundance and migratory connectivity data

The two main data inputs into the migratory network model are *abundance data* and *migratory connectivity data* for the nodes.

4.1 Abundance data

Once users have defined breeding and nonbreeding nodes for their sampled data, the species' abundance for each node is needed. In some cases, users may be using node definitions for which abundance estimates have already been calculated. Regardless of however `mignette` users obtain abundance data for the breeding and nonbreeding nodes, it needs to be formatted in the following manner for using with `mignette` as shown with the example American Redstart data:

```
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

where the first column of the data frame is the node (i.e., population) and the

second column is the abundance. All sampled nodes from the breeding and nonbreeding range need to be present in this file.

One common form of abundance data is as a raster where each cell specifies abundance (for example eBird Status and Trends data available from the ebirdst R package). If you are working with raster abundance data and need to summarize it at the node level, there are two functions in `mignette` to facilitate that process.

1. `get_vector_abunds(populations, abunds)` - sums the abundance data for a SpatRaster object of abundance (specified by the `abunds` parameter) by the vector node delineation (specified by the `populations` parameter).
2. `get_raster_abunds(populations, abunds)` - sums the abundance data for a SpatRaster object of abundance by *weighting* values based on the raster node delineation.

4.2 Migratory connectivity data

In addition to having node abundance data, `mignette` users need to have migratory connectivity data detailing the number of individuals connecting the nodes between the two stages. Migratory connectivity data has a directionality to it - i.e., there is an "encounter" season and a "recovery" season - and we use this corresponding terminology from [Procházka et al., 2017]. In `mignette` different types of migratory connectivity data use different models based on whether the breeding or nonbreeding seasons are the "encounter" or "recovery" seasons. To account for users providing different types of connectivity data, we provide three different models in mignette: 1 = the nonbreeding season is "encounter" and the breeding season is "recovery"; 2 = the breeding season is "encounter" and the nonbreeding season is "recovery"; and 3 = connectivity data are from both model types 1 and 2. For cross-season mark-recapture data (i.e. from banding or geolocators), the encounter season is where the individual is captured and the recovery season is where they are re-captured or re-sighted or, in the case of geolocator data, inferred to have originated. For genetic data, the encounter season is nonbreeding and the recovery season is breeding (i.e., "inferred").

The American Redstart assignment is all from genetic data [DeSaix et al., 2023]:

```
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

For the `mignette` migratory network, the migratory connectivity data needs to

be formatted as above: where the first column provides the node names from the "recovery" season in the rows and the remaining columns are the "encounter" season node names. The values in the data frame all the numbers of individuals assigned from the "encounter" season to the "recovery" season (in this case, American Redstarts assigned from the nonbreeding sampling location to the breeding population).

Thus, when `mignette` users have migratory connectivity data types that include assignment in both directions, they will need to create two separate data frames for those data.

Now on to Step 3) Modeling the migratory network

Chapter 5

Modeling the migratory network

5.1 Input data

As described in the previous section the data required are:

- Abundance
- Migratory connectivity

The abundance data need to be in the following format, with node IDs (same names as in the *assignment* file) in the first column and abundance values in the second column.

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 migratory connectivity data, 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

If you skipped it, see the previous Step 2) Abundance and migratory connectivity data section for specific details on the input data formatting!

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

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

For the American Redstart migratory network, we use `model = 1` from `mignette` which specifies that nonbreeding nodes are "encountered" and breeding nodes are "recovered" (i.e., inferred). This output saves the model as `amre.genetic.model_1.txt`. Below we specify `parallel = TRUE` to run MCMC on multiple cores and use the remaining defaults described previously CHANGE LINK. This step is computationally intensive and takes ~2 minutes to run on a 2023 MacBook Pro with an Apple M2 Pro chip.

```
network_model <- run_network_model(abundance = amre_abundance,
                                   nb2br_assign = amre_assign,
                                   brnode_names = brnode_names,
                                   nbnode_names = nbnode_names,
                                   model = "BR", base_filename = "amre.genetic",
                                   parallel = TRUE)
```

The `run_network_model()` function outputs a list object with four components. The first component of the output, `[["conn"]]`, is an R tibble object of the mean connectivity estimated between nodes (Table 1). These values are interpreted as the proportion of individuals the global population that migrate between the corresponding populations, as such all of the values in the network matrix sum to one. The second component, `[["jags_out"]]`, is the full output from `jagsUI::autojags()` provided as a list object, which contains important model information such as parameter estimates and credible intervals, model specifications, and goodness of fit. The final two components, `[["brnode_names"]]` and `[["nbnode_names"]]`, store the node names corresponding to the rows and columns, respectively, of the connectivity matrix.

```
network_model$conn
```

Breeding	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

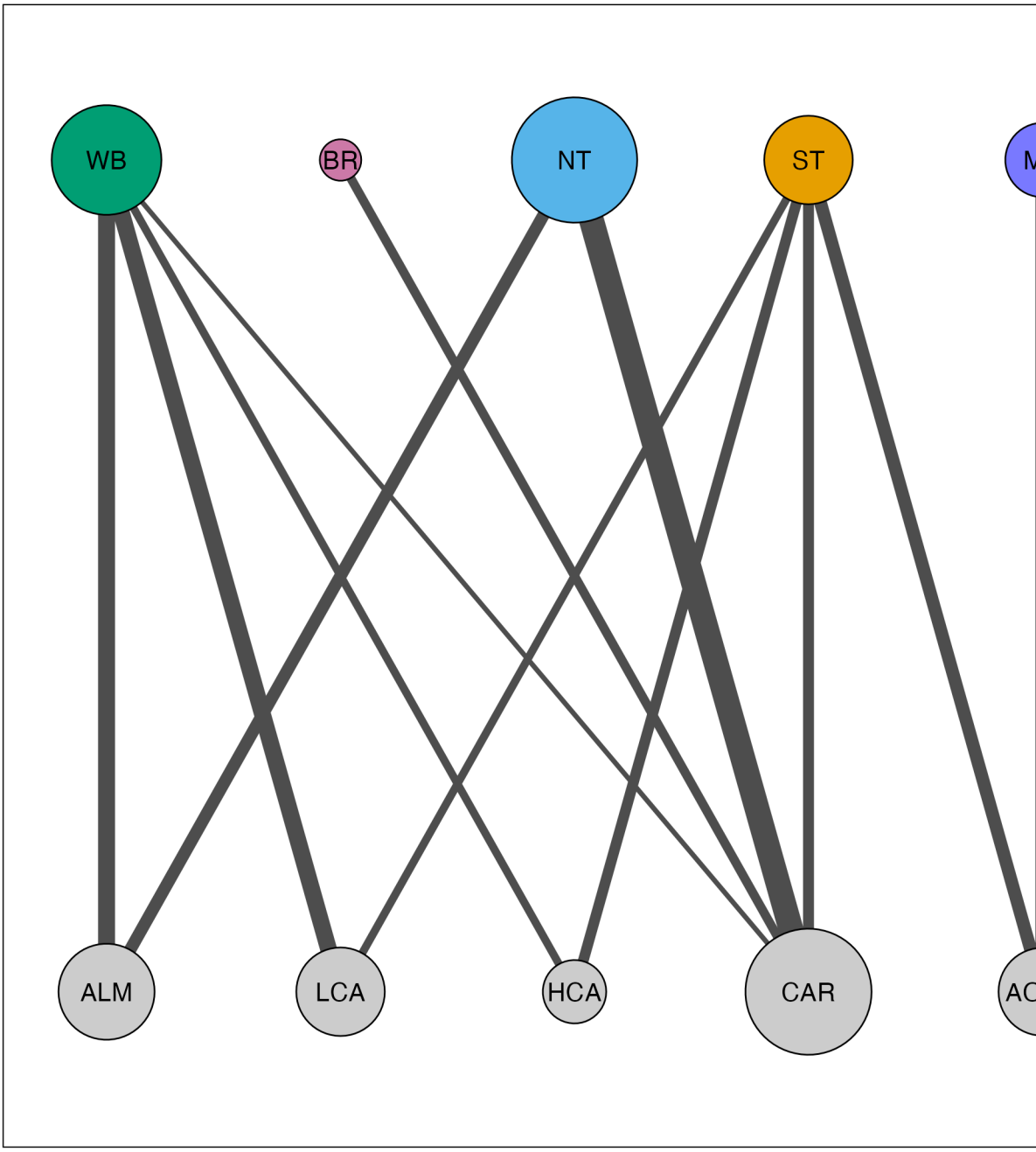
The second component is the full output from `*jagsUI* autojags()` and is accessed by `network_model$jags_out`. Here, the raw connectivity matrix can be accessed:

```
amre_conn <- network_model$jags_out$mean$conn_g
amre_conn
```

0.14212	0.13901	0.01346	0.00507	0.00002
0.00010	0.00021	0.00008	0.02082	0.00004
0.05680	0.00041	0.00013	0.38276	0.00006
0.00003	0.01345	0.03618	0.03902	0.06453
0.00013	0.00036	0.00022	0.00002	0.08496

The raw connectivity matrix is used to plot the network. We plot the migratory network with the provided `mignette` functions `net_create()` and `net_draw()`. We set `connected_tol = 0.01` which plots only the edges with connectivity values of greater than 0.01. We also demonstrate below how to change parameters such as the colors and the node and edge scale size.

```
amre_net <- net_create(network_model = network_model,
                       margin = 0.05)
#set the display size range for nodes (min and max), default 1-10
amre_net$display_par$node_size_scale <- c(8,25)
#set the display size range for edges (min and max), default 1-10
amre_net$display_par$edge_size_scale <- c(1,5)
# change colors
amre_net$display_par$brnode_colors <- c("#009e73", "#cc79a7", "#56b4e9", "#e69f00", "#7979ff")
amre_net$display_par$nbnode_colors <- "grey80"

net_draw(amre_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. Breeding populations are in the top row, for which we provided

custom colors, and nonbreeding populations are in the bottom row.

Now you have a migratory network! Check out the visualization supplement for additional ideas on plotting the network.

Chapter 6

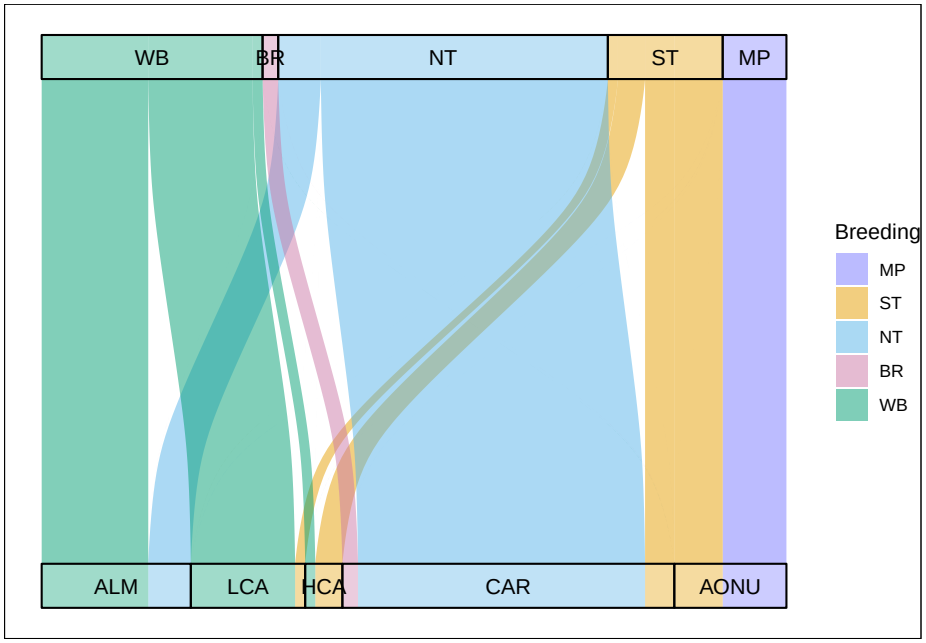
Supplement: Visualizations

6.1 Alluvial plot

Ok, we'll toss out another way to visualize these networks using the `ggalluvial` package because it's really fun. In this alluvial plot, the width of the connections correspond to the network model connectivity output, thus correspond to the proportion of the global species abundance that use that migratory network. The bars on the top row are the breeding nodes scaled by abundance and the bottom row are the nonbreeding nodes scaled by abundance.

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

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

cluster_colors <-  c(
  `ST` = "#e69f00", # orange/Southern Temperate
  `BR` = "#cc79a7", # pink/Basin Rockies (BR)
  `NT` = "#56b4e9", # light blue/Northern Temperate (NT)
  `WB` = "#009e73", # green/Western Boreal (WB)
  `MP` = "#7979ff" # dark blue/Maritime Provinces (MP)
)

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(brnode_names)),
```

```
        Nonbreeding = factor(Nonbreeding, levels = rev(nbnode_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
```



Chapter 7

Supplement: Avian abundance data and genetic breeding nodes

This supplementary section provides more detailed information outside the scope of `mignette`. Specifically, the code here goes into details of using avian abundance data and genetic populations from a genoscape.

7.1 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.

7.1.1 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).

7.1.2 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.

```
library(mignette)
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")
breeding_range <- terra::vect(breeding_range)

breeding_range_st <- sf::st_as_sf(breeding_range) # tess3Q_map_rasters function requires sf objec
```

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

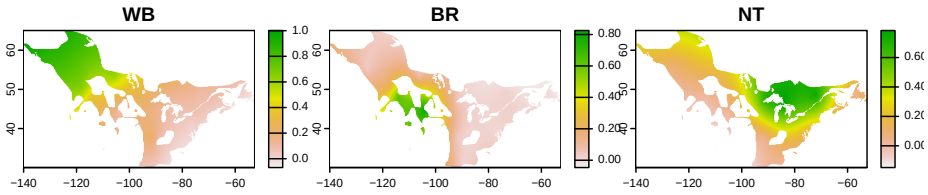
```
cluster_colors <-  c(
  `ST` = "#e69f00", # orange/Southern Temperate
  `BR` = "#cc79a7", # pink/Basin Rockies (BR)
  `NT` = "#56b4e9", # light blue/Northern Temperate (NT)
  `WB` = "#009e73", # green/Western Boreal (WB)
  `MP` = "#7979ff" # dark blue/Maritime Provinces (MP)
)
```

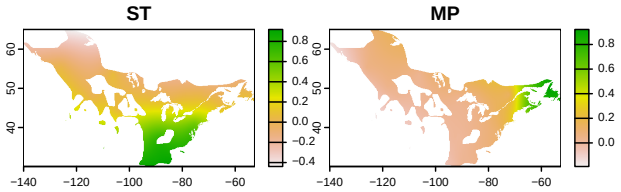
We will use a modified version of the `tess3r` package to create the genoscape rasters.

```
# 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.

7.1.3 Genoscape polygons

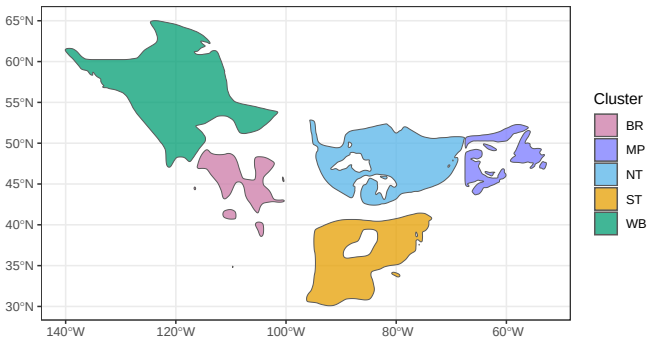
Using the genoscape rasters we will convert them to polygons, using the handy `scape_to_shape()` function. The `prob_threshold` parameter specifies the value to determine if a raster cell is included in the polygon for that genoscape. This value should be customized for different species to check for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


65°N
60°N
55°N
50°N
45°N
40°N
35°N
30°N
140°W
120°W
100°W
80°W
60°W
Cluster
BR
MP
NT
ST
WB

Bibliography

Matthew G DeSaix, Eric C Anderson, Christen M Bossu, Christine E Rayne, Teia M Schweizer, Nicholas J Bayly, Darshan S Narang, Julie C Hagelin, H Lisle Gibbs, James F Saracco, et al. Low-coverage whole genome sequencing for highly accurate population assignment: Mapping migratory connectivity in the american redstart (setophaga ruticilla). *Molecular Ecology*, 2023.

Petr Procházka, Steffen Hahn, Simon Rolland, Henk van der Jeugd, Tibor Csörgő, Frédéric Jiguet, Tomasz Mokwa, Felix Liechti, Didier Vangeluwe, and Fränzi Korner-Nievergelt. Delineating large-scale migratory connectivity of reed warblers using integrated multistate models. *Diversity and Distributions*, 23(1):27–40, 2017.

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.