A Quick Introduction to bbsBayes

Adam C. Smith, and Brandon P.M. Edwards

2021-08-11

Contents

Chapter 1

Customized BBS trends, trajectories, graphs, and maps using the bbsBayes R-package, an Introduction

1.1 What is the workshop about?

This is a 2-hour introductory workshop/demonstration of the R-package bbsBayes (https://github.com/BrandonEdwards/bbsBayes). This package allows anyone to apply the hierarchical Bayesian models used to estimate status and trends from the North American Breeding Bird Survey. The package also lets the user generate a suite of alternative metrics using the existing model output from the annual CWS analyses.

1.2 Intended Audience

Everyone is welcome! Some familiarity working with R is required if you'd also like to run the code yourself during the workshop, and the relevant packages should be installed beforehand (details will be provided with a full outline to be provided closer to the date). Those who might particularly be interested in attending include:

1. Anyone who might wish for a BBS trend for a particular region or time-period (e.g,. I want 15-year trends not 10-year trends, I wish I had a trend for just the western portion of the Barn Swallow's range),

2. Anyone who might want to analyse the BBS with a customized model (e.g., the effect of weather, land-cover, etc. on trends). Do not worry if Bayesian analyses are new to you. We will not review the basics of Bayesian statistical analyses, but the package can be very useful without understanding those basics.

1.3 What we hope you will learn here

- Download the raw data
- Choose and run a model, geographic stratification, and species (we will not cover formal model-selection, just explore the different model-options)
- Estimate trends for any time-period (1970-2019, 2005-2015, 1980-2000, etc.)
- Graph population trajectories
- Generate heat maps of population trends
- Create geofaceted trajectory plots
- Estimate trends for customized regions (e.g., All of the eastern Boreal, Great Plains vs eastern populations of grassland birds)

Chapter 2

Preparation and Installation

2.1 Pre-Workshop preparation

We will present this initial workshop primarily as a demonstration. So you do not need to run any actual code, in order to get a good sense of what the package can do.

If you are reasonably familiar with using R, and you do want to follow along with the coding during the workshop, please ensure you have installed the most recent version of R (4.0.3) and RStudio.

In addition, please install the bbsBayes package and download the data.

Finally, if you haven't used JAGS before, you'll also need to install this stand-alone program (the Bayesian MCMC software that the package relies on). Follow the link below to download the installer.

2.2 Installation

bbsBayes is on CRAN! There are two ways to install the package:

Option 1: Stable release from CRAN

```
# To install from CRAN:
install.packages("bbsBayes")
```

Option 2: Less-stable development version

```
# To install the development version from GitHub:
install.packages("devtools")
library(devtools)
devtools::install_github("BrandonEdwards/bbsBayes")
```

2.3 Data Retrieval

You can download BBS data by running `fetch_bbs_data`. This will save the data to a package-specific directory on your computer. You must agree to the terms and conditions of the data usage before the download will run [type yes at the prompt]. You only need run this function once for each annual update of the BBS database.

```
fetch_bbs_data()
```

There are options to download the available stop-level data, so you can use the package to access these data. However, for now the bbsBayes modeling functions only work with the route-level summaries. So use the defaults in the function.

2.4 Install JAGS

The modeling functions of bbsBayes require an installation of JAGS. JAGS is installed from SourceForge, an open software distribution agent. Follow this link to install [https://sourceforge.net/projects/mcmc-jags/].

2.5 Example data for Barns Swallow

This link allows you to download some example model results for Barn Swallow. This model output can be used with many of the visualization and estimation functions for the package (without having to wait hours-days for the models to run). To use these saved results, save these four folders to a relevant folder in your working directory. Each folder contains the input data object `jags_data` and the model output object `jags_mod`, stored as a saved R workspace. Each of the four folders contains the results from fitting one of the bbsBayes models to data for Barn Swallow. You can explore the differences among the four models in section 4.2.

Chapter 3

Stratification

3.1 Strata

All of the models supported by bbsBayes expect the BBS route-level data to be grouped into geographic strata. These strata are subregions of the survey area, and allow the parameters that model abundance and trend to vary across a species' range.

Use the `stratify()` function to group the BBS data into one of the included stratifications. Select the stratification using the argument `by` to stratify by the following options:

- bbs_cws – `stratify(by = "bbs_cws")` Political region X Bird Conservation region intersection (Canadian Wildlife Service [CWS] method). Identical to bbs_usgs, except all routes in BCR 7 are combined into a single stratum, and all routes in PEI and Nova Scotia are combined into a single stratum.

- bbs_usgs – `stratify(by = "bbs_usgs")` Political region X Bird Conservation region intersection (United Status Geological Survey [USGS] method)

- bcr – `stratify(by = "bcr")` Bird Conservation Region only

- state – `stratify(by = "state")` Political Region only

- latlong – `stratify(by = "latlong")` Degree blocks (1 degree of latitude X 1 degree of longitude)

Now that we know the built-in stratification options, let's stratify our copy of the BBS data. Here, we will choose to stratify our data using the "bbs_cws" method:

```
# set up for the annual CWS status and trend analysis
strat_data <- stratify(by = "bbs_cws")
```

The stratified_data object includes 3 dataframes.

- `strat_data$species_strat` is a table of all species included in the dataset. It has columns for species common names in English, French, and Spanish, as well as information on the order, family, genus, species, and the numerical species indicators used in the BBS database in both character format (with leading 0s) and integer format.

```
knitr::kable(head(strat_data$species_strat[,c(3,4,5,8,9,10)]))
```

english	french	spanish	g
Black-bellied Whistling-Duck	Dendrocygne à ventre noir	Dendrocygna autumnalis	I
Fulvous Whistling-Duck	Dendrocygne fauve	Dendrocygna bicolor	I
Emperor Goose	Oie empereur	Anser canagicus	A
Snow Goose (all forms)	Oie des neiges (toutes les formes)	Anser caerulescens	A
(Blue Goose) Snow Goose	Oie des neiges (forme bleue)	Anser caerulescens (blue form)	A
Ross's Goose	Oie de Ross	Anser rossii	A

- `strat_data$bird_strat` is a very large table of each observed count for a given species X route X year combination. This is the table that includes all of the non-zero observations in the BBS database.

```
knitr::kable(head(strat_data$bird_strat[,c(1,2,6,7,14,16,17)]))
```

statenum	Route	Year	AOU	SpeciesTotal	rt.uni	rt.uni.y
11	1	1997	5880	28	11-1	11-1-1997
11	1	1997	5671	6	11-1	11-1-1997
11	1	1997	6850	3	11-1	11-1-1997
11	1	1991	6882	22	11-1	11-1-1991
11	1	2012	3160	1	11-1	11-1-2012
11	1	1991	4860	9	11-1	11-1-1991

- `strat_data$route_strat` is a large table of each BBS survey conducted. It includes a single row for each survey-event (i.e., a route X year X observer combination). It also includes the information on the weather conditions, timing, date, etc.

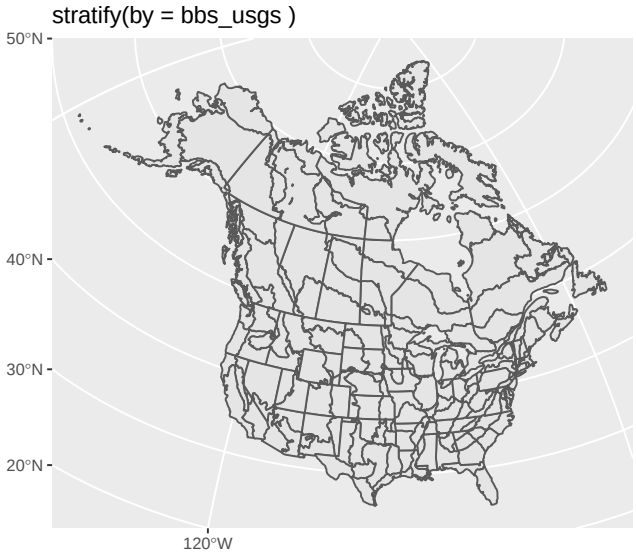
```
knitr::kable(head(strat_data$route_strat[,c(2,3,8,13,33:35)]))
```

statenum	Route	BCR	Year	strat_name	rt.uni	rt.uni.y
11	1	5	1995	CA-BC-5	11-1	11-1-1995
11	1	5	1994	CA-BC-5	11-1	11-1-1994
11	1	5	1976	CA-BC-5	11-1	11-1-1976
11	1	5	1977	CA-BC-5	11-1	11-1-1977
11	1	5	1974	CA-BC-5	11-1	11-1-1974
11	1	5	1975	CA-BC-5	11-1	11-1-1975

3.1.1 Accessing the strata maps

The package includes a function to directly access the strata maps (other than the `generate_map()` function, which displays a map of estimated trends). However this code below should suffice to find the map files in your local installation.

```
library(sf)
library(ggplot2)
laea = st_crs("+proj=laea +lat_0=40 +lon_0=-95") # Lambert equal area coordinate reference syste


stratifications <- c("bbs_cws","bbs_usgs","bcr","latlong","state")

strata_map = load_map(stratifications[2]) #selecting the map for "bbs_usgs" stratification
strata_map = st_transform(strata_map,crs = laea) #transforming the coordinate reference system
# mapping using geom_sf()
st_gg = ggplot(data = strata_map)+
geom_sf()+
labs(title = paste("stratify(by =",stratifications[2],")"))

plot(st_gg)
```


stratify(by = bbs_usgs)
50°N
40°N
30°N
20°N
120°W

Chapter 4

Prepare the data for a model and species

4.1 Prepare data

The `prepare_jags_data()` function is used to select a species, a model, the time-scale, etc.

The critical arguments are:

- strat_data: the list of stratified data returned by `stratify()`
- species_to_run: the English name for the species. This is currently case sensitive, and must exactly match one of the names in the english column of the species_strat object of strat_data i.e., `strat_data$species_strat$english` If you'd like to export a full list of the available species, this code will work:

```
write.csv(strat_data$species_strat,"Full_BBS_speciesList.csv")
```

Also, if you'd like to select a species using the French or Spanish names, this should work:

```
espece = "Colibri à gorge rubis"
species = strat_data$species_strat$english[which(strat_data$species_strat$french == espece)]
species
#> [1] "Ruby-throated Hummingbird"

especies = "Archilochus colubris"
species = strat_data$species_strat$english[which(strat_data$species_strat$spanish == especies)]
```

```
species
#> [1] "Ruby-throated Hummingbird"
```

- model: the model to use, options are "slope," "firstdiff," "gam," or "gamye"

There are additional, optional arguments:

- heavy_tailed - Logical indicating whether the extra-Poisson error distribution should be modeled as a t-distribution, with heavier tails than the standard normal distribution. Default is currently FALSE, but recent research suggest users should strongly consider setting this to TRUE, even though it requires much longer convergence times
- n_knots - Number of knots to be used in GAM function
- min_year - Minimum year to keep in analysis
- max_year - Maximum year to keep in analysis
- min_n_routes - Minimum routes per strata where species has been observed. Defaults to 3, but consider lowering to 1 if using the "latlong" stratification
- min_max_route_years - Minimum number of years with non-zero observations of species on at least 1 route. Defaults to 3
- min_mean_route_years - Minimum average of years per route with the species observed. Defaults to 1.

4.2 Models

The package has (currently) four status and trend models that differ in the way they model the time-series of observations. The four model options are slope, gam, gamye, and firstdiff.

4.2.1 slope

The slope option estimates the time series as a log-linear regression with random year-effect terms that allow the trajectory to depart from the smooth regression line. It is the model used by the USGS and CWS to estimate BBS trends between 2011 and 2018. The basic model was first described in 2002 (Link and Sauer 2002a) and its application to the annual status and trend estimates is documented in (Sauer and Link 2011) and (Smith et al. 2014).

4.2.2 gam

The gam option models the time series as a semiparametric smooth using a Generalized Additive Model (GAM) structure that shares information among the strata on the shape of the population trajectory. This model is described in (Smith and Edwards 2020).

4.2.3 gamye

The gamye option includes the semiparametric smooth used in the gam option, but also includes random year-effect terms that track annual fluctuations around the smooth, also in (Smith and Edwards 2020). This is the model that the Canadian Wildlife Service is now using for the annual status and trend estimates.

4.2.4 firstdiff

The firstdiff option models the time-series as a random-walk from the first year, so that the first-differences of the sequence of year-effects are random effects with mean = 0 and an estimated variance. This model has been described in (Link, Sauer, and Niven, n.d.). The first difference model is unique among these four in that it does not share information across the strata on the rate of population change. The population trajectories in each stratum are estimated independently of all other strata.

4.3 CWS analysis example

The annual CWS analysis uses the heavy-tailed version of the GAMYE model. Here's the data function call that is used in the 2019 analysis for Barn Swallow:

```
jags_data <- prepare_jags_data(strat_data = strat_data,
                  species_to_run = "Barn Swallow",
                  model = "gamye",
                  min_max_route_years = 2,
                  heavy_tailed = TRUE)
```

Chapter 5

Running the model

After stratifying and preparing the data for a particular species and model, the `run_model()` function calls the JAGS program to fit the selected model to the species' data.

Depending on the model and the size of the dataset (number of counts, number of strata, number of years, etc.), this process can take a long time (hours - days). The following code shows the settings used for most species in the annual CWS analysis.

```
# Not Run - Note: for Barn Swallow, this may require ~ 6-days to run!
jags_mod <- run_model(jags_data = jags_data,
                      n_iter = 24000, #higher than the default 10,000
                      n_burnin = 20000,
                      n_chains = 3,
                      n_thin = 20, #saves memory by retaining only 1/20 posterior samples
                      parallel = TRUE,
                      inits = NULL,
                      parameters_to_save = c("n","n3"))
```

There is only one critical argument in the `run_model()` function.

- jags_data - List output from the `prepare_jags_data()` function.

The remaining arguments allow for some customization of the MCMC process. The default settings are sufficient for many species and model combinations, but here is some explanation of the arguments that many users may want to modify:

- parameters_to_save - Character vector of parameters to monitor in JAGS. Defaults to just monitoring "n," the annual indices of abundance that make up the estimated population trajectories in each stratum. In the two models with separate parameters for the year-effects and the long-term change ("gamye" and "slope"), there is an optional version of the annual indices of abundance called "n3," which tracks the population trajectory after removing the year-effects. This "n3" parameter represents the smooth-only population trajectory from the gamye and the linear-slope component of the trajectory from the slope model. In the CWS annual analyses, the "n3" parameter is used to estimate trends and the "n" parameter is used to track the full population trajectory.
- n_chains - Optional number of chains to run. Defaults to 3.
- n_burnin - Optional integer specifying the number of iterations to burn in the model. Defaults to 20000 per chain.
- n_thin - Optional number of steps to thin or discard. Defaults to 10
- n_iter - Optional number of iterations per chain. Defaults to 10000.
- parallel - Logical, Should each chain be run in parallel on separate cores? If TRUE, the number of cores used will be the minimum of the n_chains specified and the number of cores on your computer

5.1 Pacfic Wren example

Here's a toy example using data for the Pacific Wren and the slope model for a shortened time-series:

```
#strat_data <- stratify(by = "bbs_cws")

# Prepare the stratified data for use in a JAGS model (see Chapter 4).
jags_data <- prepare_jags_data(strat_data = strat_data,
                               species_to_run = "Pacific Wren",
                               model = "slope",
                               min_year = 2009,
                               max_year = 2018)

# Now run the model
jags_mod <- run_model(jags_data = jags_data,
                      n_adapt = 100,
                      n_burnin = 200,
                      n_iter = 200,
                      n_thin = 1,
                      parameters_to_save = c("n","beta","strata"))
```

5.2 Convergence of the MCMC

The `run_model()` function will send a warning if Gelman-Rubin Rhat cross-chain convergence criterion is > 1.1 for any of the monitored parameters. Re-running the model with a longer burn-in and/or more posterior iterations or greater thinning rates may improve convergence. We can check for ourselves the effective sample size for each monitored parameter, as well as the Rhat values for each monitored parameter:

```
jags_mod$n.eff #shows the effective sample size for each monitored parameter
jags_mod$Rhat # shows the Rhat values for each monitored parameter
```

The seriousness of these convergence failures is something the user must interpret for themselves. In some cases some parameters of the model may not be separately estimable, but if there is no direct inference drawn from those separate parameters, their convergence may not be necessary. If all or the vast majority of the n parameters have converged (e.g., you're receiving this warning message for other monitored parameters), then inference on population trajectories and trends from the model are reliable.

If important monitored parameters have not converged, the shinystan package has some wonderful interactive tools for better understanding convergence issues with MCMC output.

```
#install.packages("shinystan")
library(shinystan)
my_sso <- shinystan::launch_shinystan(shinystan::as.shinystan(jags_mod$samples, model_name = "my_
```

bbsBayes also includes a function to help re-start an MCMC chain, so that you avoid having to wait for an additional burn-in period.

```
### if jags_mod has failed to converge...
new_initials <- get_final_values(jags_mod)

jags_mod2 <- run_model(jags_data = jags_data,
                       n_adapt = 0,
                       n_burnin = 0,
                       n_iter = 200,
                       n_thin = 1,
                       inits = new_initials,
                       parameters_to_save = c("n","beta","strata"))
```

Chapter 6

Estimating Annual Indices and Trends

Once the model has finished running, we can use the saved output to calculate the population trajectories and trends. As an example, we'll load some saved model output for Barn Swallow, from the gamye model. If you haven't already, you can download the model output file here, then save it into your working directory.

This saved workspace includes two objects:

1. jags_data - the data object created with the `prepare_jags_data()` function.
2. jags_mod - the model output created with the `run_model()` function.

```
load("data/gamye_Barn_Swallow/jags_mod_full.RData")
```

6.1 Annual Indices

The annual indices of relative abundance ("annual indices") from all of the bbsBayes models represent the expected mean count on the BBS routes in a given region and year. The time-series of these annual indices in a given region make up the estimated population trajectory.

```
indices <- generate_indices(jags_mod = jags_mod,
                            jags_data = jags_data)
## Note: this function can take ~20 minutes to run for a species with a broad range (many strata)
```

By default, this function generates estimates for the continent (i.e., survey-wide) and for the individual strata. However, if using one of the two BBS stratifications "bbs_usgs" or "bbs_cws," the user can also select summaries for composite regions (regions made up of collections of strata), such as countries, provinces/states, Bird Conservation Regions, etc.

```
indices <- generate_indices(jags_mod = jags_mod,
                            jags_data = jags_data,
                            regions = c("continental",
                            "national",
                            "prov_state",
                            "stratum"))
                            #also "bcr", #BCR specific estimates
                            # "bcr_by_country" #e.g., splits the BCRs along the nation
```

The `generate_indices()` function creates a list that includes an object called data_summary, which is a dataframe with all of the estimated annual indices `indices$data_summary`.

```
knitr::kable(head(indices$data_summary[,c(1,2,4,7,8,13,14,15,16)]))
```

Year	Region	Region_type	Index	Index_q_0.025	Index_q_0.975	obs_mean	nrts
1966	Continental	continental	9.48	8.64	10.8	2.36	396
1967	Continental	continental	9.78	9.05	10.8	4.87	814
1968	Continental	continental	10.23	9.59	11.1	7.64	869
1969	Continental	continental	10.66	10.10	11.4	8.75	1187
1970	Continental	continental	11.09	10.57	11.7	9.45	1276
1971	Continental	continental	11.36	10.89	11.9	9.84	1368

For example, one may wish to export the annual indices to a text file that can be read using Microsoft Excel.

```
write.csv(indices$data_summary, "Barn_Swallow_trajectories.csv")
```

The list also includes a number of other objects that retain the stratification information, strata area weights, and the full posterior samples for every estimated annual index. Generally these other objects are only used by subsequent functions and the user won't access them directly.

The function also accepts the alternative annual indices, such as the smooth-only indices from the gamye model.

```
indices_smooth <- generate_indices(jags_mod = jags_mod,
                            jags_data = jags_data,
                            alternate_n = "n3",
```

```
                              regions = c("continental",
                              "national",
                              "prov_state",
                              "stratum"))
                              #also "bcr", #BCR specific estimates
                              # "bcr_by_country" #e.g., splits the BCRs along the national borders
```

6.2 Population Trends

Population trends can be calculated from the output of `generate_indices()`. The trends are expressed as geometric mean rates of change (%/year) between two points in time. $Trend = 100 * ((\frac{n[Maxyear]}{n[Minyear]})^{\frac{1}{(Maxyear-Minyear)}}) - 1)$

```
trends <- generate_trends(indices = indices,
                          Min_year = 1970,
                          Max_year = 2019)
```

The `generate_trends()` function returns a dataframe with 1 row for each unit of the region-types requested (i.e., 1 row for each stratum, 1 continental, etc.). The dataframe has at least 27 columns that report useful information related to each trend, including the start and end year of the trend, lists of included strata, total number of routes, number of strata, mean observed counts, and estimates of the % change in the population between the start and end years.

```
knitr::kable(head(trends[,c(1,3,8,9,14)]))
```

Start_year	Region	Trend	Trend_Q0.025	Trend_Q0.975
1970	Continental	-0.816	-0.941	-0.699
1970	CA	-2.419	-2.736	-2.115
1970	US	-0.314	-0.427	-0.213
1970	AB	-1.961	-2.723	-1.257
1970	AK	-5.451	-8.247	-2.611
1970	AL	2.439	1.805	3.053

The `generate_trends` function includes some other arguments that allow the user to adjust the quantiles used to summarize uncertainty (e.g., interquartile range of the trend estiamtes, or the 67% CIs), as well as include additional calculations, such as the probability a population has declined (or increased) by $>$ X%.

```
trends_10 <- generate_trends(indices = indices,
                          Min_year = 2009,
                          Max_year = 2019,
```

```
                            prob_decrease = c(30,50),
                            prob_increase = c(0))
```

```
knitr::kable(head(trends[,c(1,4,8,15,22)]))
```

Start_year	Region_alt	Trend	Percent_Change	Relative_Abundance
1970	Continental	-0.816	-33.1	10.95
1970	Canada	-2.419	-69.9	6.35
1970	United States of America	-0.314	-14.3	13.85
1970	Alberta	-1.961	-62.1	6.72
1970	ALASKA	-5.451	-93.6	1.94
1970	ALABAMA	2.439	225.7	18.58

And trends calculated from the smooth component can be derived from the related annual indices.

```
trends_smooth <- generate_trends(indices = indices_smooth,
                            Min_year = 1970,
                            Max_year = 2019,
                            prob_decrease = c(30,50),
                            prob_increase = c(0))

trends_smooth_10 <- generate_trends(indices = indices_smooth,
                            Min_year = 2009,
                            Max_year = 2019,
                            prob_decrease = c(30,50),
                            prob_increase = c(0))
```

6.3 Alternative time-spans for trends

It is simple to estimate trends for alternative intervals of time. For example, calculating a 15 year trend for Barn Swallow.

```
trends_smooth_15 <- generate_trends(indices = indices_smooth,
                            Min_year = (2019-15),
                            Max_year = 2019,
                            prob_decrease = c(30,50),
                            prob_increase = c(0))
```

Or, a 15-year trend for the previous 15 years, i.e., the interval from 1989-2004:

```
trends_smooth_15alt <- generate_trends(indices = indices_smooth,
                           Min_year = 1989,
                           Max_year = 2004,
                           prob_decrease = c(30,50),
                           prob_increase = c(0))
```

6.4 Alternative definitions of trends

The end-point trend definition introduced by (Link and Sauer 2002b) is only one possible metric to measure a population trend. The default trend calculation is an interval-specific estimate of the geometric mean annual change in the population. $Trend = 100 * ((\frac{n[Maxyear]}{n[Minyear]})^{\frac{1}{(Maxyear-Minyear)}} - 1)$ It relies on a comparison of the annual indices in the first and last years of the trend period to quantify the mean rate of population change. However, it ignores the pattern of change between the two end-points.

The user can choose an alternative estimate of change that is calculated by fitting a log-linear slope to the series of all annual indices between the two end-points (e.g., all 11 years in a 10-year trend from 2008-2018). The slope of this line could be expressed as an average annual percent change across the time-period of interest. If working with estimates derived from a model with strong annual fluctuations and for which no decomposition is possible (e.g., "firstdiff" model), this slope-based trend may be a more comprehensive measure of the average population change, that is less dependent on the particular end-point years. These slope trends can be added to the trend output table by setting the `slope = TRUE` argument in `generate_trends()`. The standard trends are still calculated, but additional columns are added that include the alternate estimates. NOTE: the `generate_map()` function can map slope trends as well with the same `slope = TRUE` argument.

For example, we could use the full annual indices from the gamye model (indices including the annual fluctuations) to calculate both the default end-point trends and the alternative slope trends.

```
trends_15_slope <- generate_trends(indices = indices,
                           Min_year = (2019-15),
                           slope = TRUE,
                           Max_year = 2019,
                           prob_decrease = c(30,50),
                           prob_increase = c(0))
```

Setting `slope = TRUE` adds the slope-trend calculation to the output dataframe. Note: The probabilities of decrease and increase values are not based on the

slope metric, they are always a function of the difference in population between the start and end of the time-interval.

```
knitr::kable(head(trends_15_slope[,c(1,4,8,22)]))
```

Start_year	Region_alt	Trend	Slope_Trend
2004	Continental	-1.262	-1.282
2004	Canada	-0.524	-0.274
2004	United States of America	-1.386	-1.443
2004	Alberta	-1.832	-1.750
2004	ALASKA	-4.449	-4.828
2004	ALABAMA	-2.652	-2.414

6.4.1 Percent Change and probability of change

The `generate_trends()` function also produces estimates of the overall percent-change in the population between the first and last years of the trend-period. This calculation is often easier to interpret than an average annual rate of change. These percent change estimates have associated uncertainty bounds, and so can be helpful for deriving statements such as "between 2008 and 2018, the population has declined by 20 percent, but that estimate is relatively uncertain and the true decline may be as little as 2 percent or as much as 50 percent"

In addition, the function can optionally calculate the posterior conditional probability that a population has changed by at least a certain amount, using the `prob_decrease` and `prob_increase` arguments. These values can be useful for deriving statements such as "our model suggests that there is a 95% probability that the species has increased (i.e., $> 0\%$ increase) and a 45 percent probability that the species has increased more than 2-fold (i.e., $> 100\%$ increase)"

```
knitr::kable(head(trends_smooth_15[,c(1,4,8,15,22)]))
```

Start_year	Region_alt	Trend	Percent_Change	prob_decrease_30_percent
2004	Continental	-1.119	-15.53	0.000
2004	Canada	-0.315	-4.62	0.000
2004	United States of America	-1.255	-17.26	0.000
2004	Alberta	-1.854	-24.47	0.246
2004	ALASKA	-4.551	-50.28	0.806
2004	ALABAMA	-2.481	-31.40	0.597

6.5 Changing trends through time

Changes in populations, both annual rates of change (trends) and overall changes in populations (percent change), are often used to prioritize species

in conservation assessments. For example, the International Union for the Conservation of Nature (IUCN) or the Committee on the Status of Endangered Wildlife in Canada (COSEWIC) use estimates of population change over three generations to classify species as Endangered or Threatened. Of course the estimate of population trend used to classify a species is at least partly dependent on the year in which the species is assessed. Populations rarely follow constant rates of log-linear change through time, and so estimates of change also change through time. To quantify and visualize these changes in trends, one can estimate a series of trends for a rolling window of time-intervals. As an example, the annual CWS analysis includes an estimate of all possible 10-year trends for each species, which we refer to as the rolling trends analysis. Conceptually, it is relatively simple, and just involves estimating the annual indices for the full time-series of the survey, and then embedding the `generate_trends()` function within a loop.

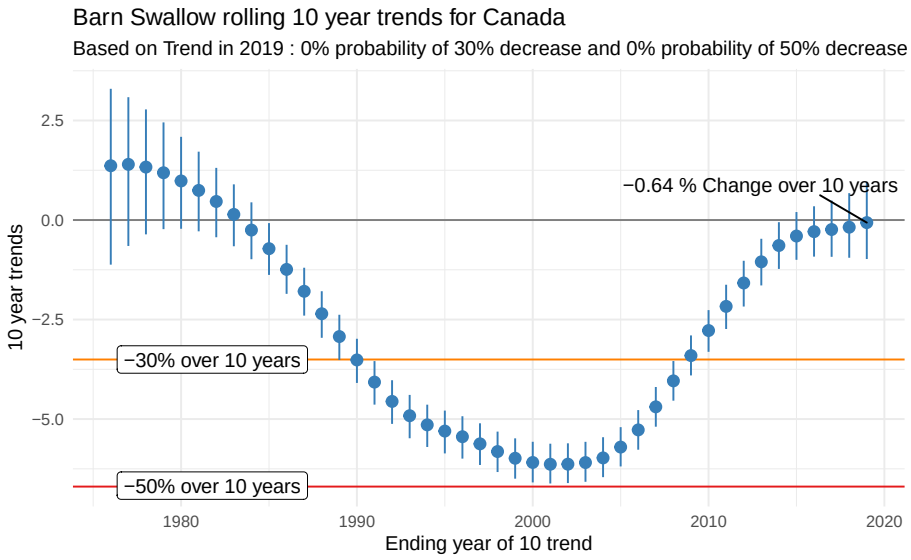
```
library(tidyverse)

indices_smooth_national <- generate_indices(jags_mod = jags_mod,
                            jags_data = jags_data,
                            alternate_n = "n3", # the smooth only component of the gamye model
                            regions = c("continental",
                            "national")) #only continental and national estimates


first_year <- min(indices_smooth_national$data_summary$Year) #First year in the time-series
last_year <- max(indices_smooth_national$data_summary$Year) #Last year in the time-series
trend_time <- 10 #use 10-year trends

trends_out <- NULL

for(Y in first_year:(last_year-trend_time)){

trends_temp <- generate_trends(indices = indices_smooth_national,
                           Min_year = Y,
                           Max_year = Y+trend_time,
                           prob_decrease = c(30,50),
                           prob_increase = c(0))

    trends_out = rbind(trends_out,trends_temp)


}



knitr::kable(head(trends_out[,c(1,2,4,8,22)]))
```

Start_year	End_year	Region_alt	Trend	prob_decrease_30_percent
1966	1976	Continental	3.54	0.000
1966	1976	Canada	1.36	0.002
1966	1976	United States of America	4.72	0.000
1967	1977	Continental	3.42	0.000
1967	1977	Canada	1.40	0.001
1967	1977	United States of America	4.47	0.000

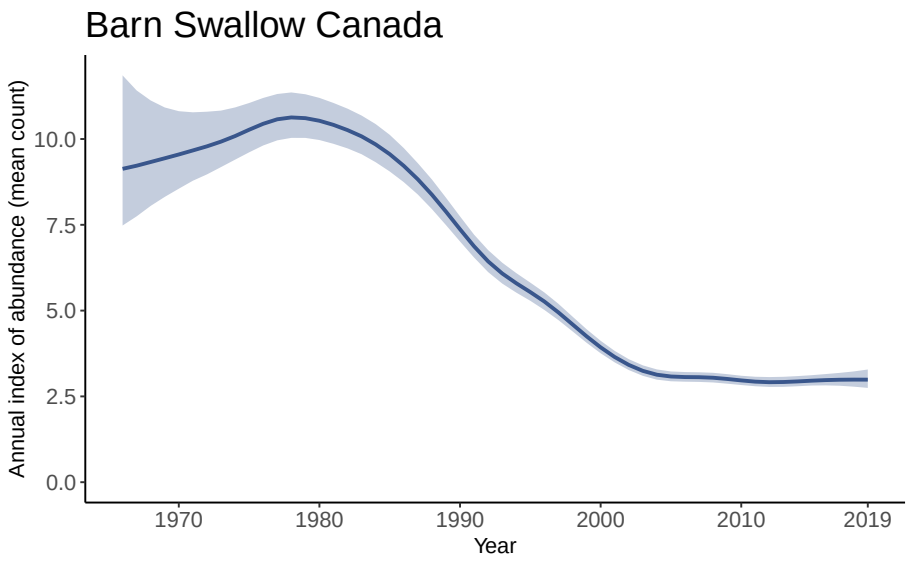
Now the `trends_out` object is a dataframe with all of the trend estimates, which we could export to a file or use to plot the estimated trends through time.

```
library(ggrepel) #handy ggplot labeling extension
library(RColorBrewer) #handy colour palettes
c_orng = RColorBrewer::brewer.pal(9,"Set1")[5]
c_red = RColorBrewer::brewer.pal(9,"Set1")[1]
c_blue = RColorBrewer::brewer.pal(9,"Set1")[2]



  # selecting just the trends for Canada
    trends_canada = trends_out[which(trends_out$Region_alt == "Canada"),]
  # adding column names to identify which credible intervals to plot below
  # also adding labels that are ggplot friendly (they do not include ".")
    trends_canada$lci = trends_canada$Trend_Q0.025
    trends_canada$uci = trends_canada$Trend_Q0.975



   # pulling out one row of the data frame to help annotate the plot
    trends_canadaend4 = trends_canada[nrow(trends_canada)-4,]
    trends_canadaend4$lab50 = "50% CI"
    trends_canadaend4$lab95 = "95% CI"

    trends_canadaend = trends_canada[nrow(trends_canada),]

    pth_30_labs = paste0(signif(100*trends_canadaend[,"prob_decrease_30_percent"],2),"%
    pth_50_labs = paste0(signif(100*trends_canadaend[,"prob_decrease_50_percent"],2),"%
    trends_canadaend$pdec = paste(signif(trends_canadaend[,"Percent_Change"],2),"% Char

  # a few extra objects to further annotate the plot
  thresh30 = (0.7^(1/trend_time)-1)*100 # threshold trend value for a 30% decline over
  thresh50 = (0.5^(1/trend_time)-1)*100 # threshold trend value for a 50% decline over

  threshs = data.frame(thresh = c(thresh30,thresh50),
```

```
                      p_thresh = c(paste("-30% over",trend_time,"years"),
                                   paste("-50% over",trend_time,"years")),
                      Year = rep(min(trends_canada$End_year),2))


  cpt = ggplot(data = trends_canada,aes(x = End_year,y = Trend))+
    theme_minimal()+
    labs(title = paste("Barn Swallow rolling",trend_time,"year trends for Canada"),
         subtitle = paste("Based on Trend in",last_year,":",pth_30_labs,"and",pth_50_labs))+
         xlab(paste("Ending year of",trend_time,"trend"))+
         ylab(paste(trend_time,"year trends"))+
    geom_hline(yintercept = thresh30,colour = c_orng)+
    geom_hline(yintercept = thresh50,colour = c_red)+
    geom_hline(yintercept = 0,colour = grey(0.5))+
    geom_label_repel(data = threshs,aes(x = Year,y = thresh,label = p_thresh),position = "nudge
    geom_pointrange(aes(x = End_year,y = Trend,ymin = lci,ymax = uci),colour = c_blue)+
    geom_text_repel(data = trends_canadaend,aes(x = End_year,y = Trend,label = pdec),min.segmen


print(cpt)
```



And here's the population trajectory for comparison. Note: the `plot_indices()` function that generated this plot is demonstrated in the next section 7.



Chapter 7

Trajectory graphs and trend maps

Now that we have calculated values for the trends and trajectories, bbsBayes also includes functions to plot and map the values.

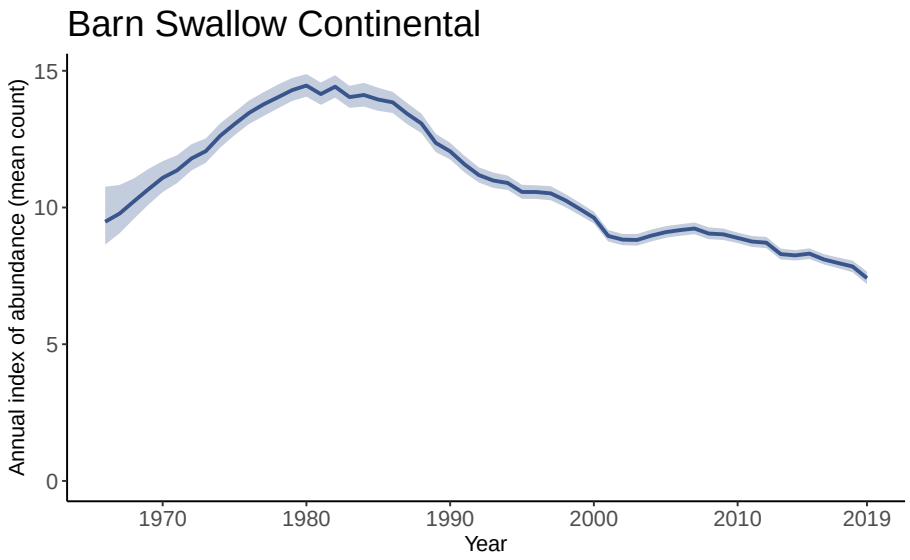
7.1 Graphing the trajectories (annual indices)

The `plot_indices()` function produces a list of ggplot figures that can be combined into a single pdf file:

```
# Not run
tp = plot_indices(indices = indices,
                          species = "Barn Swallow",
                  title_size = 20,
                  axis_title_size = 12,
                  axis_text_size = 12)
pdf(file = "Barn Swallow Trajectories.pdf")
print(tp)
dev.off()
```

Or, we can print plots to individual devices:

```
tp = plot_indices(indices = indices,
                          species = "Barn Swallow",
                  title_size = 20,
                  axis_title_size = 12,
                  axis_text_size = 12)
print(tp[[1]])
```



`plot_indices()` also allows the user to add points to show the observed mean counts as well as stacked dots to indicate the number of observations in each year.

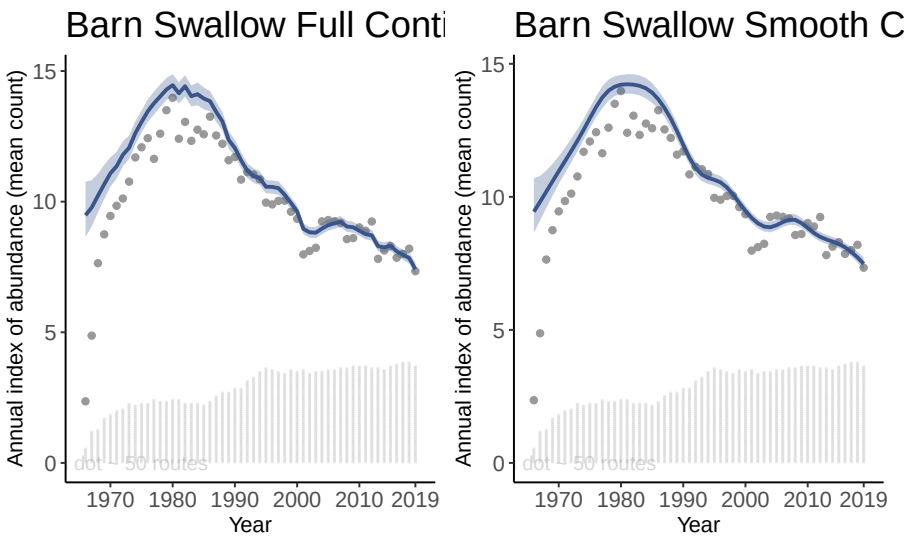
```
tp2 = plot_indices(indices = indices,
                           species = paste("Barn Swallow","Full"),
                   add_observed_means = TRUE,
                   add_number_routes = TRUE,
                   title_size = 20,
                   axis_title_size = 12,
                   axis_text_size = 12)
```

And we can compare these indices plots with those generated using only the smooths.

```
tp3 = plot_indices(indices = indices_smooth,
                           species = paste("Barn Swallow","Smooth"),
                   add_observed_means = TRUE,
                   add_number_routes = TRUE,
                   title_size = 20,
                   axis_title_size = 12,
                   axis_text_size = 12)
```

Using the patchwork library, we can show these plots side-by-side:

```
library(patchwork)
print(tp2[[1]]+tp3[[1]])
```



7.2 Adding elements to indices plots

The output of the `plot_indices()` function is a list of ggplot objects, it is relatively easy to add components to the basic plot.

Here, we'll use the first plot (continental trajectory) from the full population trajectory version above `tp2[[1]]` as the base plot, and we'll add the smooth population trajectory information to the graph in a different colour.

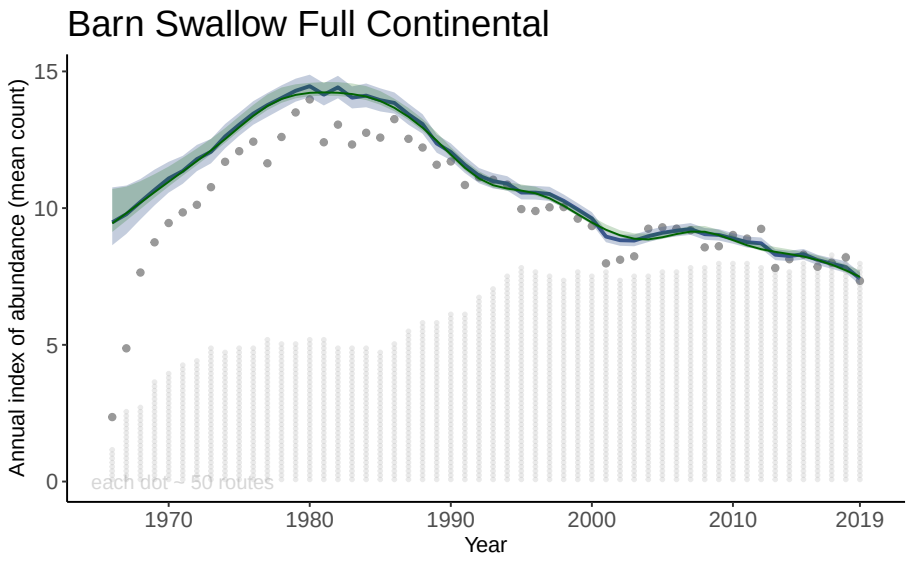
First, extract the smooth population trajectory information in a dataframe.

```
library(tidyverse)
continental_smooth <- indices_smooth$data_summary %>%
  filter(Region == "Continental")
```

Then we overlap the original plot with a new line and uncertainty ribbon.

```
overlay_plot <- tp2[[1]] +
  geom_ribbon(data = continental_smooth,inherit.aes = FALSE,
```

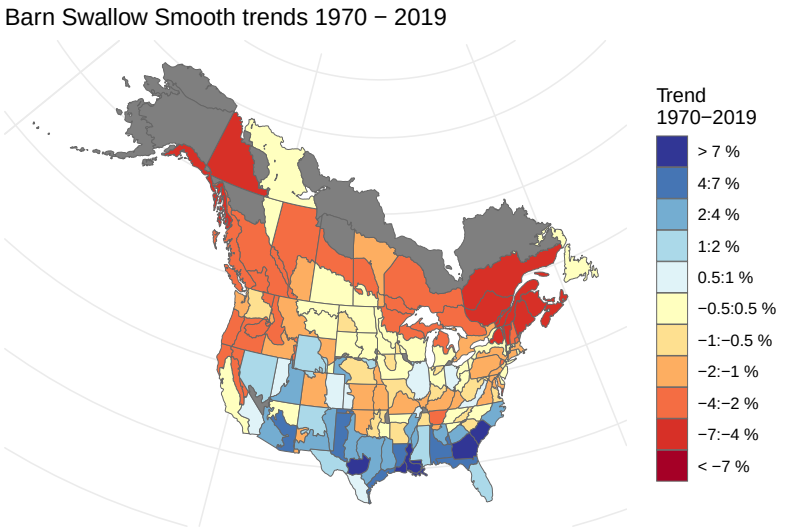
```
              aes(x = Year,ymax = Index_q_0.25,ymin = Index_q_0.975),
              alpha = 0.2,
              fill = "darkgreen")+
  geom_line(data = continental_smooth, inherit.aes = FALSE,
            aes(x = Year,y = Index),
            colour = "darkgreen")
print(overlay_plot)
```



7.3 Mapping the trends

The trends can be mapped to produce strata maps coloured by species population trends.

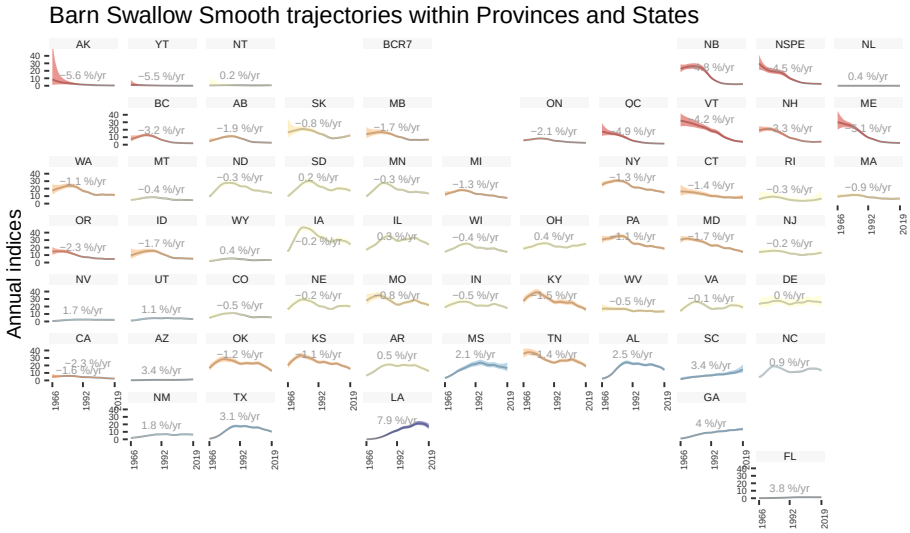
```
mp = generate_map(trends_smooth,
                  select = TRUE,
                  stratify_by = "bbs_cws",
                  species = "Barn Swallow Smooth")
print(mp)
```



7.4 Geofacet Trajectories

For stratifications that can be compiled by political regions (i.e., `bbs_cws`, `bbs_usgs`, or `state`), the function `geofacet_plot` will generate a ggplot object that plots the state and province level population trajectories in facets arranged in an approximately geographic arrangement. These plots offer a concise, range-wide summary of a species' population trajectory.

```
gf <- geofacet_plot(indices_list = indices_smooth,
                    select = TRUE,
                    stratify_by = "bbs_cws",
                    multiple = FALSE,
                    trends = trends_smooth,
                    slope = F,
                    species = "Barn Swallow Smooth")

print(gf)
```


Barn Swallow Smooth trajectories within Provinces and States
Annual indices
AK
−5.6 %/yr
YT
−5.5 %/yr
NT
0.2 %/yr
BCR7
NB
−3.8 %/yr
NSPE
−4.5 %/yr
NL
0.4 %/yr
BC
−3.2 %/yr
AB
−1.9 %/yr
SK
−0.8 %/yr
MB
−1.7 %/yr
ON
−2.1 %/yr
QC
−4.9 %/yr
VT
−4.2 %/yr
NH
−3.3 %/yr
ME
−1.1 %/yr
WA
−1.1 %/yr
MT
−0.4 %/yr
ND
−0.3 %/yr
SD
0.2 %/yr
MN
−0.3 %/yr
MI
−1.3 %/yr
NY
−1.3 %/yr
CT
−1.4 %/yr
RI
−0.3 %/yr
MA
−0.9 %/yr
OR
−2.3 %/yr
ID
−1.7 %/yr
WY
0.4 %/yr
IA
−0.2 %/yr
IL
0.3 %/yr
WI
−0.4 %/yr
OH
0.4 %/yr
PA
−1 %/yr
MD
−1.7 %/yr
NJ
−0.2 %/yr
NV
1.7 %/yr
UT
1.1 %/yr
CO
−0.5 %/yr
NE
−0.2 %/yr
MO
−0.8 %/yr
IN
−0.5 %/yr
KY
−1.5 %/yr
WV
−0.5 %/yr
VA
−0.1 %/yr
DE
0 %/yr
CA
−2.3 %/yr
−1.6 %/yr
AZ
3.4 %/yr
OK
−1.2 %/yr
KS
−1.1 %/yr
AR
0.5 %/yr
MS
2.1 %/yr
TN
−1.4 %/yr
AL
2.5 %/yr
SC
3.4 %/yr
NC
0.9 %/yr
NM
1.8 %/yr
TX
3.1 %/yr
LA
7.9 %/yr
GA
4 %/yr
FL
3.8 %/yr
1966
1992
2019

Chapter 8

Advanced options

8.1 Custom regional summaries

Yes, you can calculate the trend and trajectories for custom combinations of strata, such as a formal trend estimate for populations of Barn Swallow in the South East portion of their range (e.g., BCRs 21,24,25,26,27,35,36, and 37).

8.1.1 Define the custom regions as a collection of the existing strata.

First extract a dataframe that defines the original strata used in the analysis.

```
st_comp_regions <- get_composite_regions(strata_type = "bbs_cws")
knitr::kable(head(st_comp_regions))
```

prov_state	region	area_sq_km	national	bcr	Province_State	Country	bcr_by_
AB	CA-AB-10	52565	CA	10	Alberta	Canada	Canada-
AB	CA-AB-6	445136	CA	6	Alberta	Canada	Canada-
AB	CA-AB-8	6987	CA	8	Alberta	Canada	Canada-
AB	CA-AB-11	149352	CA	11	Alberta	Canada	Canada-
AK	US-AK-1	9551	US	1	ALASKA	United States of America	United S
AK	US-AK-2	283405	US	2	ALASKA	United States of America	United S

Then add a column to the dataframe that groups the original strata into the desired custom regions.

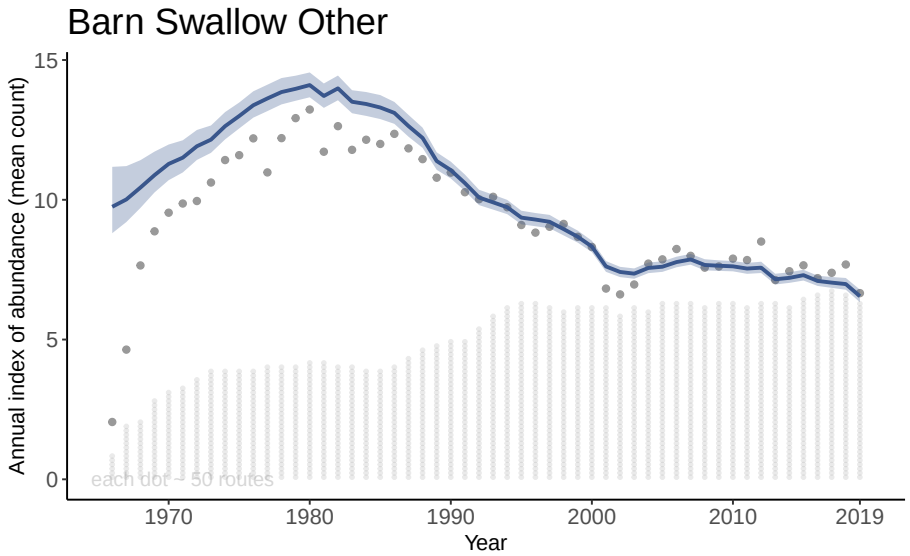
```
st_comp_regions$SouthEast <- ifelse(st_comp_regions$bcr %in% c(21,24:27,35:37),"SouthEast","Other
```

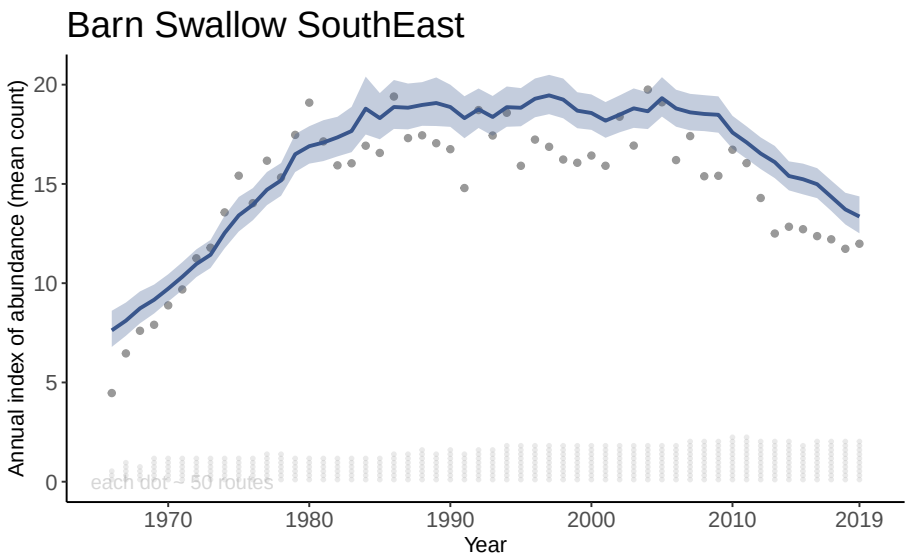
8.1.2 Use defined regions to estimate indices and trends

st_comp_regions can now be used as the dataframe input to the argument alt_region_names in `generate_indices()`, with "SouthEast" as the value for the argument regions. The relevant trends can be calculated using just the `generate_trends()` function.

```
library(patchwork)
custom_indices <- generate_indices(jags_mod = jags_mod,
                                   jags_data = jags_data,
                                   alt_region_names = st_comp_regions,
                                   regions = "SouthEast")
tp = plot_indices(indices = custom_indices,
                  species = "Barn Swallow",
                  add_observed_means = TRUE,
                  add_number_routes = TRUE,
  title_size = 20,
  axis_title_size = 12,
  axis_text_size = 12)
```

```
print(tp[[1]])
print(tp[[2]])
```





Calculate trends

```
custom_trends <- generate_trends(indices = custom_indices)
knitr::kable(custom_trends[,c(1,3,8,9,14)])
```

8.2 Exporting the JAGS model

You can easily export any of the bbsBayes models to a text file.

```
model_to_file(model = "slope",
              filename = "my_slope_model.txt")
```

This is the best way to confirm all of the priors, and model structures.

8.3 Customizing the JAGS model and data

You can also modify the model text (e.g., try a different prior) and run the modified model

```
run_model <- function(... ,
                      model_file_path = "my_modified_slope_model.txt",
                      ... )
```

Or, you could add covariates. For example a GAM smooth to estimate the effect of the day of year on the observations, or an annual weather covariate, or... Then add the relevant covariate data to the jags_data object, and you're off!

Here's a GitHub repo with an example of a modified `bbsBayes` model. The modified model includes a series of GAM smooths to account for variation in counts over the course of the BBS observation season (mid-May through early July). In addition these seasonal smooths are allowed to vary by decade to account for possible shifts in phenology through time.

https://github.com/AdamCSmithCWS/BBS_seasonal_shift_ALHU

8.4 Comparing Models

Finally, bbsBayes can be used to run cross-validations. For example, the `get_final_values()` function is useful to provide an efficient starting point for a cross-validation runs, without having to wait for another full burn-in period.

Paper that includes an example of how to implement a cross-validation using bbsBayes.

Ornithological Applications: https://doi.org/10.1093/ornithapp/duaa065 Supplement: https://zenodo.org/badge/latestdoi/228419725

NOTE: although bbsBayes includes functions to calculate WAIC, recent work has shown that WAIC performs very poorly with the BBS data (https://doi.org/10.1650/CONDOR-17-1.1). We recommend a k-fold cross-validation approach, as in the above zenodo archive.

Chapter 9

References

Link, William A., and John R. Sauer. 2002a. "A Hierarchical Analysis of Population Change with Application to Cerulean Warblers." *Ecology* 83 (10): 2832–40. https://doi.org/10.1890/0012-9658(2002)083%5B2832:AHAOPC%5D2.0.CO;2.

———. 2002b. "A Hierarchical Analysis of Population Change with Application to Cerulean Warblers." *Ecology* 83 (10): 2832–40. https://doi.org/10.1890/0012-9658(2002)083%5B2832:AHAOPC%5D2.0.CO;2.

Link, William A., John R. Sauer, and Daniel K. Niven. n.d. "Model Selection for the North American Breeding Bird Survey." *Ecological Applications* n/a (n/a). https://doi.org/10.1002/eap.2137.

Sauer, John R., and William A. Link. 2011. "Analysis of the North American Breeding Bird Survey Using Hierarchical Models." *The Auk* 128 (1): 87–98. https://doi.org/10.1525/auk.2010.09220.

Smith, Adam C., and Brandon P.M. Edwards. 2020. "North American Breeding Bird Survey Status and Trend Estimates to Inform a Wide-Range of Conservation Needs, Using a Flexible Bayesian Hierarchical Generalized Additive Model." https://doi.org/10.1101/2020.03.26.010215.

Smith, Adam C., Marie-Anne R. Hudson, Constance Downes, and Charles M. Francis. 2014. "Estimating Breeding Bird Survey Trends and Annual Indices for Canada: How Do the New Hierarchical Bayesian Estimates Differ from Previous Estimates?" *The Canadian Field-Naturalist* 128 (2): 119. https://doi.org/10.22621/cfn.v128i2.1565.