

Package: bbsBayes2 (via r-universe)

October 22, 2026

Type Package

Title Hierarchical Bayesian Analysis of North American BBS Data

Version 1.2026.4

Imports backports ($\geq 1.4.1$), dplyr ($\geq 1.1.0$), ggrepel, geofacet, ggplot2 ($\geq 3.4.0$), HDInterval, magrittr, mgcv, posterior ($\geq 1.2.1$), purrr ($\geq 0.3.4$), readr ($\geq 2.1.2$), rlang ($\geq 0.4.11$), sbtools, sf ($\geq 1.0.8$), spdep ($\geq 1.2.7$), snakecase, stringr, tidyr ($\geq 1.2.0$), units ($\geq 0.8.0$), withr ($\geq 2.5.0$), cmdstanr

Depends R (≥ 4.1)

SystemRequirements CmdStan

(<https://mc-stan.org/users/interfaces/cmdstan>)

URL <https://github.com/bbsBayes/bbsBayes2>,
<https://bbsbayes.github.io/bbsBayes2/>

BugReports <https://github.com/bbsBayes/bbsBayes2/issues>

NeedsCompilation no

Description The North American Breeding Bird Survey (BBS) is a long-running program that seeks to monitor the status and trends of birds that spend the breeding season in Canada and the United States. Since its start in 1966, the BBS has accumulated almost 60 years of data for over 500 species of North American Birds. Given the temporal and spatial structure of the data, hierarchical Bayesian models are used to assess the status and trends of these 500+ species of birds. 'bbsBayes2' allows you to perform hierarchical Bayesian analysis of BBS data. You can fit 9 different models to the observations for one or more species, and you can take specify how the data should be stratified, what spatial and temporal extent of the data should be included, the settings for the sampling of the posterior distributions, access and modify the Stan models, and explore model diagnostics, including prior and posterior predictions, convergence, and measures of relative model fit (`loo` and cross-validation).

License MIT + file LICENSE
Encoding UTF-8
LazyData true
LazyDataCompression bzip2
Roxygen list(markdown = TRUE)
Suggests covr, knitr, rmarkdown, rnatualearth, rnatualearthhires,
 roxygen2, testthat (>= 3.0.0)
Additional_repositories <https://stan-dev.r-universe.dev/>,
<https://ropensci.r-universe.dev>
Config/testthat/edition 3
VignetteBuilder knitr
Config/roxygen2/version 8.0.0
Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libsecret-
 1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev
Repository <https://bbsbayes.r-universe.dev>
Date/Publication 2026-09-22 15:40:49 UTC
RemoteUrl <https://github.com/bbsBayes/bbsBayes2>
RemoteRef HEAD
RemoteSha 3d89663886b7f705e1d7b6f8ac583c72f57582a1

Contents

assign_prov_state	3
bbs_data_sample	4
bbs_models	5
bbs_strata	6
bbsBayes2-defunct	6
bbsBayes2-deprecated	7
copy_model_file	7
equal_area_crs	8
fetch_bbs_data	9
generate_indices	10
generate_trends	14
get_convergence	18
get_model_vars	19
get_summary	19
have_bbs_data	20
have_cmdstan	21
load_bbs_data	22
load_map	23
pacific_wren_model	24
plot_geofacet	24

assign_prov_state 3

plot_indices	26
plot_map	28
prepare_data	30
prepare_model	32
prepare_spatial	35
remove_cache	37
run_model	38
save_model_run	41
search_species	42
species_forms	43
species_notes	44
stratify	44

Index 49

assign_prov_state	<i>Categorize polygon by Province/State</i>
--------------------------	---

Description

Categorizes custom stratification polygons by province or state if possible. This can be useful for calculating regional indices (`generate_indices()`) or trends (`generate_trends()`) on a custom stratification, or if you want to create a geofaceted plot (`plot_geofacet()`).

Usage

```
assign_prov_state(  
  strata_map,  
  min_overlap = 0.75,  
  plot = FALSE,  
  keep_spatial = TRUE  
)
```

Arguments

strata_map	sf data frame. Strata polygons to be categorized.
min_overlap	Numeric. The minimum proportion of overlap between a stratum polygon and a Province or State. Overlap below this proportion will raise warnings.
plot	Logical. Whether to plot how polygons were assigned to Provinces or States
keep_spatial	Logical. Whether the output should be a spatial data frame or not.

Value

(Spatial) data frame with strata assigned to Province/State.

See Also

Other helper functions: `load_map()`, `search_species()`

Examples

```
# Demonstration of why we can't divide BCR by Provinces and States!
map <- load_map("bcr")
assign_prov_state(map, plot = TRUE)

# Use custom stratification, using sf map object
# e.g. with WBPBS stratum boundaries 2019
# available: https://ecos.fws.gov/ServCat/Reference/Profile/142628

## Not run:
map <- sf::read_sf("../WBPBS_Stratum_Boundaries_2019") %>%
  rename(strata_name = STRAT) # expects this column
s <- assign_prov_state(map, plot = TRUE)
# Some don't divide nicely, we could try a different min_overlap

s <- assign_prov_state(map, min_overlap = 0.6, plot = TRUE)

## End(Not run)
```

bbs_data_sample	<i>Sample BBS data</i>
-----------------	------------------------

Description

Contains only Pacific Wren data

Usage

```
bbs_data_sample
```

Format

bbs_data_sample:

A list containing:

- **birds** - counts of each bird seen per route per
- **routes** - data for each route run per year
- **species** - species list of North America

Details

A sample dataset containing only data for Pacific Wrens for the 2025 state-level BBS data. The full count set is obtained via the function `fetch_bbs_data(release = 2025)`. The data are obtained from the BBS database and is subject to change as new data are added each year. See References for citation.

Source

<https://www.sciencebase.gov/> via `fetch_bbs_data()`

References

Ziolkowski Jr., D.J., Lutmerding, M., Skalos, S.M., English, W.B., and Hudson, M-A.R., 2025, North American Breeding Bird Survey Dataset 1966 - 2024: U.S. Geological Survey data release, <https://doi.org/10.5066/P14SNUV4>.

bbs_models

Stan models included in bbsBayes2

Description

These models are included in bbsBayes2. The model files themselves can be found in the folder identified by `system.file("models", package = "bbsBayes2")`. To create a custom Stan model, see `copy_model_file()` and the `model_file` argument of `prepare_model()`. See also the [models article](#) for more details.

Usage

bbs_models

Format

bbs_models:

A data frame with 9 rows and 3 columns:

- `model` - Model type
 - `first_diff` - First difference models
 - `gam` - General Additive Models (GAM)
 - `gamy` - General Additive Models (GAM) with Year Effect
 - `slope` - Slope models
- `variant` - Variant of the model to run
 - `nonhier` - Non-hierarchical models (only available for first difference models)
 - `hier` - Hierarchical models
 - `spatial` - Spatial models
- `file` - Stan model file name

Examples

bbs_models

bbs_strata	List of included strata
Description List of strata included in bbsBayes2. Each list item contains a data frame describing the strata for that stratification (name, area, country, etc.) Usage bbs_strata Format bbs_strata: A list of 6 data frames Contains bbs - New as of version 1.1.3.1 Intersection of States-Provinces-Territories with the updated (as of 2025) Bird Conservation Regions (BCRs). Updated BCRs include changes made to the Northern BCRs. BCR "numbers" are now alpha-numeric to account for new regions that are subdivisions of previous numbered BCRs. For example, BCRs 6S and 6N are southern and northern subregions of the previous BCR 6. bbs_usgs - Intersection of States-Provinces-Territories with the previous version of the Bird Conservation Regions (BCRs) bbs_cws- Same as bbs_usgs, with two modifications to suit Canadian context: Intersection of States-Provinces-Territories with the previous version of the Bird Conservation Regions (BCRs), and BCR 7 is treated as a single stratum (pooled across all provinces and territories), and Prince Edward Island is combined with Nova Scotia - pooled across the two provinces bcr - Updated (2025) version of the Bird Conservation Regions (BCRs). latlong - 1-degree latitude * longitude grid. The design stratification of the BBS. prov_state - Canadian provinces/territories and states of the continental United States of America - excludes Hawaii (there are no BBS routes in HI) Examples bbs_strata[["bbs_cws"]] bbs_strata[["latlon"]]	
bbsBayes2-defunct	bbsBayes2 defunct functions

Description

Superseded:

No superseded functions

No longer relevant:

No non-relevant functions

Arguments

... Original function arguments

See Also

[bbsBayes2-deprecated](#)

bbsBayes2-deprecated	<i>bbsBayes2 deprecated functions</i>
----------------------	---------------------------------------

Description

No deprecated functions

Arguments

... Original function arguments

See Also

[bbsBayes2-defunct](#)

copy_model_file	<i>Copy model file</i>
-----------------	------------------------

Description

Save a predefined Stan model file to a local text file for editing. These files can then be used in `prepare_model()` by specifying the `model_file` argument.

Usage

```
copy_model_file(model, model_variant, dir, overwrite = FALSE)
```

Arguments

- | | |
|----------------------------|--|
| <code>model</code> | Character. Type of model to use, must be one of "first_diff" (First Differences), "gam" (General Additive Model), "gamye" (General Additive Model with Year Effect), or "slope" (Slope model). |
| <code>model_variant</code> | Character. Model variant to use, must be one of "nonhier" (Non-hierarchical), "hier" (Hierarchical; default), or "spatial" (Spatially explicit). |
| <code>dir</code> | Character. Directory where file should be saved. |
| <code>overwrite</code> | Logical. Whether to overwrite an existing copy of the model file. |

Value

File path to copy of the new model file.

See Also

Other modelling functions: `run_model()`, `save_model_run()`

Examples

```
# Save the Slope model in temp directory
copy_model_file(model = "slope", model_variant = "spatial", dir = tempdir())

# Overwrite an existing copy
copy_model_file(model = "slope", model_variant = "spatial", dir = tempdir(),
                overwrite = TRUE)

# Clean up
unlink(file.path(tempdir(), "slope_spatial_bbs_CV_COPY.stan"))
```

`equal_area_crs`

equal_area_crs

Description

Introduced in version 1.1.2. The specific coordinate reference system (crs) used for the included strata maps. `North_America_Albers_Equal_Area_Conic`, projection that provides good visualisation of the main areas covered by the North American Breeding Bird Survey

Usage

```
equal_area_crs
```

Format

```
equal_area_crs:
  An object of class crs
```

Examples

```
equal_area_crs
```

fetch_bbs_data

Fetch Breeding Bird Survey dataset

Description

Fetch and download Breeding Bird Survey data from the United States Geological Survey (USGS) FTP site. This is the raw data that is uploaded to the site before any analyses are performed. Users can download different types (**state**, **stop**) and different releases (currently 2020, and annually for years 2022 through 2026).

Usage

```
fetch_bbs_data(
  level = "state",
  release = 2026,
  force = FALSE,
  quiet = FALSE,
  compression = "none",
  include_unacceptable = FALSE,
  interactive = TRUE
)
```

Arguments

level	Character. Which type of BBS counts to use, "state" or "stop". Default "state".
release	Numeric. Which yearly release to use, 2022 (including data through 2021 field season) or 2020 (including data through 2019). Default 2022.
force	Logical. Should pre-existing BBS data be overwritten? Default FALSE.
quiet	Logical. Suppress progress messages? Default FALSE.
compression	Character. What compression should be used to save data? Default is none which takes up the most space but is the fastest to load. Must be one of none , gz , bz2 , or xz (passed to <code>readr::write_rds()</code> 's <code>compress</code> argument).
include_unacceptable	Logical. DO NOT change this unless you are very confident that you want these data. Should the package retain the BBS data collected on non-standard BBS routes (including some with fewer than 50 stops), or outside of the acceptable weather, season, and times-of-day structure of the BBS survey design. Default FALSE for good reason. If you change this to TRUE and thereby decide you want to include the BBS data that do not fit the BBS structured survey design, be sure you understand the information in at least the following columns of the route survey info: RouteTypeID, RouteTypeDetailID, RPID, RunType, and the rest of the information in the data release meta data.

interactive Logical. Default is TRUE, to allow users to see and agree to the data terms. If set to FALSE, then the download takes place without the user's input, useful for scripted processes such as running analyses in HPC environments or building containers.

Details

Users will be asked before saving the BBS data to a package-specific directory created on their computer. Before downloading any data, users must thoroughly read through the USGS terms and conditions for that data and enter the word "yes" to agree.

BBS **state** level counts provide counts beginning in 1966, aggregated in five bins, each of which contains cumulative counts from 10 of the 50 stops along a route. In contrast, BBS **stop** level counts provides stop-level data beginning in 1997, which includes counts for each individual stop along routes. **Note that stop-level data is not currently supported by the modelling utilities in bbsBayes2.**

There are multiple releases for each type of data, 2020, and annual releases for years 2022 through 2026. By default all functions use the most recent release unless otherwise specified. For example, the **release** argument in **stratify()** can be changed to 2020 to use the 2020 release of state-level counts.

bbsBayes2 by default removes observations from routes that were not established following the stratified random design, the handful of routes that are on water (not on roadsides), that we established without using the stratified random design (e.g., within protected areas), any survey that was not conducted during acceptable survey conditions (high winds, heavy precipitation, or outside of the acceptable time windows), and surveys that were not completed (< 45 of the 50 stops were surveyed, often for unsafe survey conditions or degrading weather conditions).

See Also

Other BBS data functions: [have_bbs_data\(\)](#), [load_bbs_data\(\)](#), [remove_cache\(\)](#)

Examples

```
fetch_bbs_data(force = TRUE)
fetch_bbs_data(level = "stop", force = TRUE)
fetch_bbs_data(release = 2020, force = TRUE)
fetch_bbs_data(release = 2020, level = "stop", force = TRUE)
```

generate_indices

Regional annual indices of abundance

Description

Calculate annual indices of relative abundance by year for different regions. These indices can then be used to plot population trajectories for the species, and to estimate trends.

Usage

```
generate_indices(
  model_output = NULL,
  quantiles = c(0.025, 0.05, 0.25, 0.75, 0.95, 0.975),
  regions = c("continent", "stratum"),
  regions_index = NULL,
  alternate_n = "n",
  gam_smooths = FALSE,
  start_year = NULL,
  max_backcast = NULL,
  drop_exclude = FALSE,
  hpdi = FALSE,
  quiet = FALSE
)
```

Arguments

model_output	List. Model output generated by <code>run_model()</code> .
quantiles	Numeric. Vector of quantiles to be sampled from the posterior distribution. Default is <code>c(0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975)</code> . Note that these quantiles will be used to create confidence interval bands in <code>plot_indices()</code> and by quantiles in <code>generate_trends()</code> , so make sure you specify the ones you want to use later.
regions	Character. Which region(s) to summarize and calculate indices for. Default is "continent" and "stratum". Options also include "country", "prov_state", "bcr", and "bcr_by_country". Note that some regions only apply to specific stratifications. You can also supply a custom region that exists as a column in the <code>regions_index</code> data frame (see examples for more details).
regions_index	Data frame. Custom regions to summarize. Data frame must include all strata in the original data in one column (strata_name), and any custom regions defined as categories in other columns. See examples for more details.
alternate_n	Character. Indicating the name of the alternative annual index parameter in a model, Default is "n" which for all models represents an index of the estimated annual relative abundance, scaled as the expected mean count averaged across all BBS routes and observers in a given stratum and year. For some of the models included in <code>bbsBayes2</code> , alternatives exist that provide a partial decomposition of the time-series. For the "gamye" models, the parameter "n_smooth" represents the smooth-only version of the annual index of relative abundance (i.e., the component of the annual index estimated by the spline-based smooth of the GAM). This "n_smooth" is identical to the "n" values for the same model, but excludes the annual fluctuations. For the "gamye" models, this "n_smooth" parameter is likely the most natural parameter to use in estimating trends. A similar option exists for the "slope" models, where the parameter "n_slope" represents the component of the population trajectory estimated by the

	log-linear regression slope parameters in the model. Users should be particularly cautious about interpreting this "n_slope" values for relatively long time-series. As a continuous regression slope, it assumes interpreting it as an estimate of population trajectory and using it to generate trend estimates assumes that there is a single continuous rate of population change across the entire time-series. Biologically, this may be reasonable for 10-20 year periods, but will be less reasonable for longer time-periods.
<code>gam_smooths</code>	Logical. Whether to return an array of posterior samples of gam-based smooths through the full time-series of annual indices. These can be treated as smooth population trajectories from any model. NOTE: these gam smooths require fitting <code>mgcv::gam()</code> to every posterior draw of every possible population trajectory. As a result, <code>generate_indices</code> can require many minutes to complete if set to <code>TRUE</code> .
<code>start_year</code>	Numeric. Trim the data record before calculating annual indices.
<code>max_backcast</code>	Numeric. The number of years to back cast stratum-level estimates before the first year that species was observed on any route in that stratum. Default is <code>NULL</code> , which generates annual indices for the entire time series and ignores back-casting. CWS national estimates use a back cast of 5. Note that unless <code>drop_exclude = TRUE</code> , problematic years are only flagged, not omitted. See Details for more specifics.
<code>drop_exclude</code>	Logical. Whether or not strata that exceed the <code>max_backcast</code> threshold should be excluded from the calculations. Default is <code>FALSE</code> (regions are flagged and listed but not dropped).
<code>hpdi</code>	Logical. Should credible intervals and limits be calculated using highest posterior density intervals instead of simple quantiles of the posterior distribution. Default is <code>FALSE</code> . these intervals are often a better descriptor of skewed posterior distributions, such as the predicted mean counts that the indices represent. Note hpd intervals are not stable for small percentages of the posterior distribution, and so <code>hpdi = TRUE</code> is ignored for <code>quantiles</code> values between 0.33 and 0.67 (i.e., if the <code>quantiles</code> value defines a limit for a centered hpd interval that would include < 33% of the posterior distribution).
<code>quiet</code>	Logical. Suppress progress messages? Default <code>FALSE</code> .

Details

`max_backcast` is a way to deal with the fact that the species of interest may not appear in the data until several years after the start of the time-series `max_backcast` specifies how many years can occur before the stratum is flagged. A `max_backcast` of 5 will flag any stratum without a non-zero (or non-NA) observation within the first 5 years of the time-series. Note that records are *only* flagged unless `drop_exclude = TRUE`. If you find that the early data record is sparse and results in the exclusion of many strata, consider trimming the early years by specifying a `start_year`.

Value

A list containing

- **indices** - data frame of calculated regional annual indices of abundances
- **samples** - array of posterior draws from the model
- **meta_data** - meta data defining the analysis
- **meta_strata** - data frame listing strata meta data
- **raw_data** - data frame of summarized counts
- **gam_smooth_samples** - optional-array of posterior draws of the gam smooth

indices contains the following columns:

- **year** - Year of particular index
- **region** - Region name
- **region_type** - Type of region
- **strata_included** - Strata *potentially* included in the annual index calculations
- **strata_excluded** - Strata *potentially* excluded from the annual index calculations because they have no observations of the species in the first part of the time series, see arguments **max_backcast** and **start_year**
- **index** - Annual index of relative abundance. Expected mean count by an average BBS observer, on an average BBS route in that region, in that year. For single strata, these expected mean counts will closely match the scale of the observed mean counts (**obs_mean**), after adjusting for the estimated trends as well as variation in the routes and observers with data in any given year. For composite regions (any region larger than a single stratum), these indices also reflect the relative abundance and area-weights of the strata that make up the larger region. Due to these area and abundance weights the scale of these indices will not necessarily match the scale of the observed mean counts.
- **index_q_XXX** - Strata-weighted count index (by different quantiles)
- **obs_mean** - Mean observed annual counts of birds across all routes and all years. An alternative estimate of the average relative abundance of the species in the region and year. Differences between this and the annual indices are a function of the model. For composite regions (i.e., anything other than stratum-level estimates) this average count is calculated as an area-weighted average across all strata included
- **n_routes** - Number of BBS routes that contributed data for this species, region, and year
- **n_routes_total** - Number of BBS routes that contributed data for this species and region for all years in the selected time-series, i.e., all years since **start_year**
- **n_non_zero** - Number of BBS routes on which this species was observed (i.e., count is > 0) in this region and year
- **backcast_flag** - Approximate annual average proportion of the covered species range that is free of extrapolated population trajectories. e.g., if 1.0, data cover full time-series; if 0.75, data cover 75 percent of time-series. Only calculated if **max_backcast** != NULL.

See Also

Other indices and trends functions: [generate_trends\(\)](#), [plot_geofacet\(\)](#), [plot_indices\(\)](#), [plot_map\(\)](#)

Examples

```
# Using the example model for Pacific Wrens

# Generate the continental and stratum indices
i <- generate_indices(pacific_wren_model)

# Generate the continental and stratum indices using hpdi
i <- generate_indices(pacific_wren_model, hpdi = TRUE)

# Generate only country indices
i_nat <- generate_indices(pacific_wren_model, regions = "country")

# Use a custom region specification (dummy example)
library(dplyr)
ri <- bbs_strata[["bbs_cws"]]
ri <- mutate(ri, my_region = if_else(prov_state %in% "ON",
                                     "Ontario", "Rest"))

# Generate indices with these custom regions
i_custom <- generate_indices(
  pacific_wren_model,
  regions = c("country", "prov_state", "my_region"),
  regions_index = ri)
```

generate_trends	<i>Generate regional trends</i>
-----------------	---------------------------------

Description

Generates trends for continent and strata and optionally for countries, states/provinces, or BCRs from analyses run on the stratifications that support these composite regions. Calculates the geometric mean annual changes in population size for composite regions.

Usage

```
generate_trends(
  indices,
  min_year = NULL,
  max_year = NULL,
  quantiles = c(0.025, 0.05, 0.25, 0.75, 0.95, 0.975),
  slope = FALSE,
  gam = FALSE,
  prob_decrease = NULL,
  prob_increase = NULL,
  hpdi = FALSE,
  export_full_posterior = FALSE
)
```

Arguments

indices	List. Indices generated by <code>generate_indices()</code> .
min_year	Numeric. Minimum year to use. Default (NULL) uses first year in data.
max_year	Numeric. Maximum year to use. Default (NULL) uses first year in data.
quantiles	Numeric vector. Quantiles to be sampled from the posterior distribution. Defaults to <code>c(0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975)</code> .
slope	Logical. Whether to calculate an alternative trend metric, the slope of a log-linear regression through the annual indices. Default FALSE , which estimates the trend as the geometric mean annual rate of change between min_year and max_year . This is the end-point definition of trend that only directly incorporates information from the two years, and therefore closely tracks the annual population fluctuations in those particular years. Conceptually, this metric of trend tracks the difference between the two years. If TRUE , trend represents the slope of a linear regression through the log-transformed annual indices of abundance for all years between min_year and max_year . This definition of trend is less sensitive to the particular annual fluctuations of a given min_year and max_year . Either metric may be more or less appropriate given the user's desired inference. The appropriate choice of metric may also depend on the model and the alternate_n choice made in <code>generate_indices</code> . For example if the fitted model was one of the "gamye" alternatives, and the alternate_n = "nsmooth", then the default slope = FALSE option will represent the end-point difference of the smooth component, which already excludes the annual fluctuations and so has similar inferential properties as the slope = TRUE option from the "first_diff" model.
gam	Logical. New in version 1.1.2. Optional trend calculation using end-point trends derived from posterior distribution of GAM-based smooths of each posterior draw of the estimated annual indices. Requires the output from <code>generate_indices(., gam_smooth = TRUE)</code> .
prob_decrease	Numeric vector. Percent-decrease values for which to optionally calculate the posterior probabilities (see Details). Default is NULL (not calculated). Can range from 0 to 100.
prob_increase	Numeric vector. Percent-increase values for which to optionally calculate the posterior probabilities (see Details). Default is NULL (not calculated). Can range from 0 to Inf.
hpdi	Logical. Should credible intervals and limits be calculated using highest posterior density intervals instead of simple quantiles of the posterior distribution. Default is FALSE . these intervals are often a better descriptor of skewed posterior distributions, such as the predicted mean counts that the indices represent. Note hpd intervals are not stable for small percentages of the posterior distribution, and so hpdi = TRUE is ignored for quantiles values between 0.33 and 0.67 (i.e., if the quantiles value defines a limit for a centered hpd interval that would include < 33% of the posterior distribution).

export_full_posterior

Logical. Default FALSE. If TRUE, the output list includes a tibble that contains the full posterior distribution of every trend estimate

Details

The posterior probabilities can be calculated for a percent-decrease (**prob_decrease**) and/or percent-increase (**prob_increase**) if desired. These calculate the probability that the population has decreased/increased by at least the amount specified.

For example, a **prob_increase** = 100 would result in the calculation of the probability that the population has increased by more than 100% (i.e., doubled) over the period of the trend.

Alternatively, a **prob_decrease** = 50 would result in the calculation of the probability that the population has decreased by more than 50% (i.e., less than half of the population remains) over the period of the trend.

Value

A list containing

- **trends** - data frame of calculated population trends, one row for each region in the input **indices**
- **meta_data** - meta data defining the analysis
- **meta_strata** - data frame listing strata meta data
- **raw_data** - data frame of summarized counts
- **trends_full_posterior** - tibble of trends, region names, and full posterior distribution of every trend estimate. Can be used for formal comparisons of differences in trends between time-periods, regions, etc.

trends contains the following columns:

- **start_year** - First year of the trend
- **end_year** - Last year of the trend
- **region** - Region name
- **region_type** - Type of region
- **strata_included** - Strata *potentially* included in the annual index calculations
- **strata_excluded** - Strata *potentially* excluded from the annual index calculations because they have no observations of the species in the first part of the time series, see arguments **max_backcast** and **start_year**
- **trend** - Estimated median annual percent change over the trend time-period according to end point comparison of annual indices for the **start_year** and the **end_year**
- **trend_q_XXX** - Trend estimates by different quantiles
- **percent_change** - Median overall estimate percent change over the trend time-period
- **percent_change_q_XXX** - Percent change by different quantiles
- **slope_trend** - Estimated median annual percent change over the trend time-period, according to the slope of a linear regression through the log-transformed annual indices. (Only if **slope** = TRUE)

- `slope_trend_q_XXX` - Slope-based trend estimates by different quantiles. (Only if `slope = TRUE`)
- `width_of_95_percent_credible_interval` - Width (in percent/year) of the credible interval on the trend calculation. Calculated for the widest credible interval requested in via `quantiles`. Default is 95 percent CI (i.e., `trend_q_0.975 - trend_q_0.025`)
- `width_of_95_percent_credible_interval_slope` - Width (in percent/year) of the credible interval on the slope-based trend calculation. Calculated for the widest credible interval requested in via `quantiles`. Default is 95 percent CI (i.e., `slope_trend_q_0.975 - slope_trend_q_0.025`). (Only if `slope = TRUE`)
- `prob_decrease_XX_percent` - Proportion of the posterior distribution of `percent_change` that is below the percentage values in `prob_decrease` (if non-Null)
- `prob_increase_XX_percent` - Proportion of the posterior distribution of `percent_change` that is above the percentage values in `prob_increase` (if non-Null)
- `rel_abundance` - Mean annual index value across all years. An estimate of the average relative abundance of the species in the region. Can be interpreted as the predicted average count of the species in an average year on an average route by an average observer, for the years, routes, and observers in the existing data
- `obs_rel_abundance` - Mean observed annual count of birds across all routes and all years. An alternative estimate of the average relative abundance of the species in the region. For composite regions (i.e., anything other than stratum-level estimates) this average count is calculated as an area-weighted average across all strata included.
- `n_routes` - Number of BBS routes that contributed data for this species and region for all years in the selected time-series, i.e., all years since `start_year`
- `mean_n_routes` - Mean number of BBS routes that contributed data for this species, region, and year
- `n_strata_included` - The number of strata included in the region
- `backcast_flag` - Approximate annual average proportion of the covered species range that is free of extrapolated population trajectories. e.g., if 1.0, data cover full time-series; if 0.75, data cover 75 percent of time-series. Only calculated if `max_backcast` != NULL.

See Also

Other indices and trends functions: [generate_indices\(\)](#), [plot_geofacet\(\)](#), [plot_indices\(\)](#), [plot_map\(\)](#)

Examples

```
# Using the example model for Pacific Wrens...

# Generate the continental and stratum indices
i <- generate_indices(pacific_wren_model)

# Now, generate the trends
t <- generate_trends(i)

# Use the slope method
```

```
t <- generate_trends(i, slope = TRUE)

# Calculate probability of the population declining by 50%
t <- generate_trends(i, prob_decrease = 50)
```

get_convergence	<i>Convergence metrics</i>
-----------------	----------------------------

Description

Calculate convergence metrics for the model run. Specifically calculates bulk and tail effective sample sizes (`ess_bulk`, `ess_tail`) and R-hat (`rhat`). Returns output very similar to `get_summary()`.

Usage

```
get_convergence(model_output, variables = NULL)
```

Arguments

<code>model_output</code>	List. Model output generated by <code>run_model()</code> .
<code>variables</code>	Character vector. Specific variables (e.g., <code>"strata_raw[1]"</code>) or variable types (e.g., <code>"strata_raw"</code>) for which to calculate metrics. If <code>NULL</code> (default) all variables are returned.

Value

Data frame of convergence metrics for all model variables. Contains `variable_type`, `variable`, `ess_bulk`, `ess_tail`, and `rhat`.

See Also

Other model assessment functions: [get_model_vars\(\)](#), [get_summary\(\)](#)

Examples

```
# Temporarily suppress convergence warning for legibility
# "The ESS has been capped to avoid unstable estimates."
opts <- options(warn = -1)

# Using the example model for Pacific Wrens

get_convergence(pacific_wren_model)
get_convergence(pacific_wren_model, variables = "strata_raw")
get_convergence(pacific_wren_model, variables = "strata_raw[9]")

# Restore warnings
options(opts)
```

get_model_vars	<i>Get model variables</i>
----------------	----------------------------

Description

Returns the basic model variables and/or variable types (note that most variables have different iterations for each strata and each year).

Usage

```
get_model_vars(model_output, all = FALSE)
```

Arguments

model_output	List. Model output generated by <code>run_model()</code> .
all	Logical. Whether or not to return all , specific variables (e.g., <code>strata_raw[1]</code>) or just variable types (e.g., <code>strata_raw</code>). Defaults to <code>FALSE</code> (variable types only).

Value

A character vector of all model variable types.

See Also

Other model assessment functions: [get_convergence\(\)](#), [get_summary\(\)](#)

Examples

```
# Using the example model for Pacific Wrens...

# List variable types
get_model_vars(pacific_wren_model)

# List all variables
get_model_vars(pacific_wren_model, all = TRUE)
```

get_summary	<i>Return the cmdstanr summary</i>
-------------	------------------------------------

Description

Extract and return the model summary using `cmdstanr::summary()`.

Usage

```
get_summary(model_output, variables = NULL)
```

Arguments

model_output List. Model output generated by `run_model()`.

variables Character vector. Specific variables (e.g., `"strata_raw[1]"`) or variable types (e.g., `"strata_raw"`) for which to calculate metrics. If `NULL` (default) all variables are returned.

Value

A data frame of model summary statistics, matching the output of `cmdstanr::summary()` plus one additional column `variable_type` that identifies categories of parameters (e.g., `n` for all of the annual indices for every stratum and year).

See Also

Other model assessment functions: [get_convergence\(\)](#), [get_model_vars\(\)](#)

Examples

```
# Temporarily suppress convergence warning for legibility
# "The ESS has been capped to avoid unstable estimates."
opts <- options(warn = -1)

# Using the example model for Pacific Wrens

get_summary(pacific_wren_model)
get_summary(pacific_wren_model, variables = "strata_raw")
get_summary(pacific_wren_model, variables = "strata_raw[9]")

# Restore warnings
options(opts)
```

have_bbs_data	<i>Check whether BBS data exists locally</i>
---------------	--

Description

Use this function to check if you have the BBS data downloaded and where `bbsBayes2` is expecting to find it. If it returns `FALSE`, the data is not present; use `fetch_bbs_data()` to retrieve it.

Usage

```
have_bbs_data(level = "state", release = 2026, quiet = FALSE)
```

Arguments

level	Character. BBS data to check, one of "all", "state", or "stop". Default "state".
release	Character/Numeric. BBS data to check, one of "all", or the annual releases. Default 2024
quiet	Logical. Suppress progress messages? Default FALSE.

Value

TRUE if the data is found, FALSE otherwise

See Also

Other BBS data functions: [fetch_bbs_data\(\)](#), [load_bbs_data\(\)](#), [remove_cache\(\)](#)

Examples

```
have_bbs_data()
have_bbs_data(release = 2020)
have_bbs_data(release = "all", level = "all")
```

have_cmdstan	<i>Check if cmdstan is installed</i>
--------------	--------------------------------------

Description

Wrapper around `cmdstanr::cmdstan_version(error_on_NA = FALSE)` for quick check.

Usage

```
have_cmdstan()
```

Details

Used internally for skipping examples and tests if no cmdstan installed.

load_bbs_data	<i>Load Breeding Bird Survey data</i>
---------------	---------------------------------------

Description

Load the local, minimally processed, raw, unstratified data. The data must have been previously fetched using `fetch_bbs_data()`. This function is provided for custom explorations and is not part of the analysis workflow; `stratify()` will do the loading for you.

Usage

```
load_bbs_data(level = "state", release = 2026, sample = FALSE, quiet = TRUE)
```

Arguments

level	Character. Which type of BBS counts to use, "state" or "stop". Default "state".
release	Numeric. Which yearly release to use, 2022 (including data through 2021 field season) or 2020 (including data through 2019). Default 2022.
sample	Logical. Whether or not to use the sample data for Pacific Wrens (see <code>?bbs_data_sample</code>). Default is FALSE . If TRUE , level and release are ignored.
quiet	Logical. Suppress progress messages? Default FALSE .

Value

Large list (3 elements) consisting of:

birds	Data frame of sample bird point count data per route, per year
routes	Data frame of sample yearly route data
species	Sample list of North American bird species

See Also

Other BBS data functions: [fetch_bbs_data\(\)](#), [have_bbs_data\(\)](#), [remove_cache\(\)](#)

load_map	<i>Load a geographic strata map</i>
----------	-------------------------------------

Description

Load one of the included spatial data files (maps) as a simple features object (**sf** package)

Usage

```
load_map(stratify_by = NULL, type = "strata")
```

Arguments

stratify_by	Character. Stratification type. One of "bbs", "prov_state", "bcr", "lat-long", "bbs_cws", "bbs_usgs".
type	Character. "strata" or political map ("North America", "Canada" or "US"/"USA"/"United States of America").

Value

sf polygon object

See Also

Other helper functions: [assign_prov_state\(\)](#), [search_species\(\)](#)

Examples

```
# First, stratify the sample data
strat_map <- load_map(stratify_by = "bbs")

# simple plot of the map
plot(strat_map)

# or use ggplot2
library(ggplot2)
ggplot(data = strat_map) +
  geom_sf(aes(fill = strata_name), show.legend = FALSE)
```

`pacific_wren_model` *Example model output*

Description

Example model output from running a hierarchical first difference model on the included sample data for Pacific Wrens.

Usage

```
pacific_wren_model
```

Format

```
pacific_wren_model:
```

A list output from `run_model()` with 4 items

- `model_fit` - cmdstanr model output
- `model_data` - list of data formatted for use in Stan modelling
- `meta_data` - meta data defining the analysis
- `meta_strata` - data frame listing strata meta data
- `raw_data` - data frame of summarized counts

Examples

```
# Code to replicate:
## Not run:
pacific_wren_model <- stratify(by = "bbs_cws", sample_data = TRUE,
use_map = FALSE) %>%
  prepare_data() %>%
  prepare_model(model = "first_diff", set_seed = 111) %>%
  run_model(chains = 2, iter_sampling = 1000,
            iter_warmup = 1000,
            thin = 50, set_seed = 111)

## End(Not run)
```

`plot_geofacet` *Create geofacet plot of population trajectories by province/state*

Description

Generate a faceted plot of population trajectories by province/state. Only possible if indices created by `generate_indices()` include the `prov_state` region. All geofacet plots have one facet per state/province, so if there are multiple strata per facet, these can be plotted as separate trajectories within each facet (`multiple = TRUE`).

Usage

```
plot_geofacet(
  indices,
  ci_width = 0.95,
  multiple = FALSE,
  trends = NULL,
  slope = FALSE,
  add_observed_means = FALSE,
  col_viridis = FALSE
)
```

Arguments

indices	List. Indices generated by <code>generate_indices()</code> .
ci_width	Numeric. Quantile defining the width of the plotted credible interval. Defaults to 0.95 (lower = 0.025 and upper = 0.975). Note these quantiles need to have been precalculated in <code>generate_indices()</code> .
multiple	Logical. Whether to plot multiple strata-level trajectories within each prov/state facet. Default FALSE .
trends	List. (Optional) Output generated by <code>generate_trends()</code> . If included trajectories are coloured based on the same colour scale used in <code>plot_map</code> .
slope	Logical. If trends included, whether colours in the plot should be based on slope trends. Default FALSE .
add_observed_means	Logical. Whether to include points indicating the observed mean counts. Default FALSE . Note: scale of observed means and annual indices may not match due to imbalanced sampling among routes. Also, pattern of change in observed means through time should reflect estimated population trajectories for strata-level summaries. However, the temporal patterns may be very different between means and annual indices for composite regions (e.g., continental, state, province, or BCR) because the indices for composite regions account for the variation in area weights and variation in relative abundances among strata, and the observed mean counts do not. For strata-level assessments, these observed means can be a useful model assessment tool. For composite regions, their interpretation is more complicated.
col_viridis	Logical. Should the colour-blind-friendly "viridis" palette be used. Default FALSE .

Value

ggplot object

See Also

Other indices and trends functions: [generate_indices\(\)](#), [generate_trends\(\)](#), [plot_indices\(\)](#), [plot_map\(\)](#)

Examples

```
# Using the example model for Pacific Wrens...

# Generate indices
i <- generate_indices(pacific_wren_model,
                     regions = c("stratum", "prov_state"))

# Generate trends
t <- generate_trends(i)

# Now make the geofacet plot.
plot_geofacet(i, trends = t, multiple = TRUE)
plot_geofacet(i, trends = t, multiple = TRUE, col_viridis = TRUE)
plot_geofacet(i, multiple = TRUE)
plot_geofacet(i, trends = t, multiple = FALSE)
plot_geofacet(i, multiple = FALSE)

# With different ci_width, specify desired quantiles in indices
i <- generate_indices(pacific_wren_model,
                     regions = c("stratum", "prov_state"),
                     quantiles = c(0.005, 0.995))

plot_geofacet(i, multiple = FALSE, ci_width = 0.99)
```

plot_indices

Generate plots of index trajectories by stratum

Description

Generates the indices plot for each stratum modelled.

Usage

```
plot_indices(
  indices = NULL,
  ci_width = 0.95,
  min_year = NULL,
  max_year = NULL,
  title = TRUE,
  title_size = 20,
  axis_title_size = 18,
  axis_text_size = 16,
  line_width = 1,
  spaghetti = FALSE,
  n_spaghetti = 100,
  alpha_spaghetti = 0.2,
  add_observed_means = FALSE,
  add_number_routes = FALSE
)
```

Arguments

<code>indices</code>	List. Indices generated by <code>generate_indices()</code> .
<code>ci_width</code>	Numeric. Quantile defining the width of the plotted credible interval. Defaults to 0.95 (lower = 0.025 and upper = 0.975). Note these quantiles need to have been precalculated in <code>generate_indices()</code> .
<code>min_year</code>	Numeric. Minimum year to plot.
<code>max_year</code>	Numeric. Maximum year to plot.
<code>title</code>	Logical. Whether to include a title on the plot.
<code>title_size</code>	Numeric. Font size of plot title. Defaults to 20
<code>axis_title_size</code>	Numeric. Font size of axis titles. Defaults to 18
<code>axis_text_size</code>	Numeric. Font size of axis text. Defaults to 16
<code>line_width</code>	Numeric. Size of the trajectory line. Defaults to 1
<code>spaghetti</code>	Logical. False by default. Plotting option to visualise the uncertainty in the estimated population trajectories. Instead of plotting the trajectory as a single line with an uncertainty bound, if TRUE, then the plot shows a sample of the posterior distribution of the estimated population trajectories. E.g., 100 semi-transparent lines, each representing one posterior draw of the population trajectory.
<code>n_spaghetti</code>	Integer. 100 by default. Number of posterior draws of the population trajectory to include in the plot. Ignored if <code>spaghetti = FALSE</code> .
<code>alpha_spaghetti</code>	Numeric between 0 and 1, 0.2 by default. Alpha value - transparency of each individual population trajectory line in the spaghetti plot. Ignored if <code>spaghetti = FALSE</code> .
<code>add_observed_means</code>	Logical. Whether to include points indicating the observed mean counts. Default FALSE. Note: scale of observed means and annual indices may not match due to imbalanced sampling among routes. Also, pattern of change in observed means through time should reflect estimated population trajectories for strata-level summaries. However, the temporal patterns may be very different between means and annual indices for composite regions (e.g., continental, state, province, or BCR) because the indices for composite regions account for the variation in area weights and variation in relative abundances among strata, and the observed mean counts do not. For strata-level assessments, these observed means can be a useful model assessment tool. For composite regions, their interpretation is more complicated.
<code>add_number_routes</code>	Logical. Whether to superimpose dotplot showing the number of BBS routes included in each year. This is useful as a visual check on the relative data-density through time because in most cases the number of observations increases over time.

Value

List of ggplot2 plots, each item being a plot of a stratum's indices.

See Also

Other indices and trends functions: `generate_indices()`, `generate_trends()`, `plot_geofacet()`, `plot_map()`

Examples

```
# Using the example model for Pacific Wrens...

# Generate country, continent, and stratum indices
i <- generate_indices(model_output = pacific_wren_model,
                      regions = c("country", "continent", "stratum"))

# Now, plot_indices() will generate a list of plots for all regions
plots <- plot_indices(i)

# To view any plot, use [[i]]
plots[[1]]

names(plots)

# Suppose we wanted to access the continental plot. We could do so with
plots[["continent"]]

# You can specify to only plot a subset of years using min_year and max_year

# Plots indices from 2015 onward
p_2015_min <- plot_indices(i, min_year = 2015)
p_2015_min[["continent"]]

#Plot up indices up to the year 2017
p_2017_max <- plot_indices(i, max_year = 2017)
p_2017_max[["continent"]]

#Plot indices between 2011 and 2016
p_2011_2016 <- plot_indices(i, min_year = 2011, max_year = 2016)
p_2011_2016[["continent"]]
```

plot_map

Generate a map of trends by strata

Description

`plot_map()` allows you to generate a colour-coded map of the percent change in species trends for each strata.

Usage

```
plot_map(
  trends,
  slope = FALSE,
  title = TRUE,
  alternate_column = NULL,
  col_viridis = FALSE,
  col_ebird = FALSE,
  strata_custom = NULL,
  zoom_range = TRUE
)
```

Arguments

trends	List. Trends generated by <code>generate_trends()</code> .
slope	Logical. Whether or not to map values of the alternative trend metric (slope of a log-linear regression) if <code>slope = TRUE</code> was used in <code>generate_trends()</code> , through the annual indices. Default <code>FALSE</code> .
title	Logical. Whether or not to include a title with species. Default <code>TRUE</code> .
alternate_column	Character, Optional name of numerical column in trends dataframe to plot. If one of the columns with "trend" in the title, (e.g., <code>trend_q_0.05</code> then the colour scheme and breaks will match those used in the default trend maps)
col_viridis	Logical. Should the colour-blind-friendly "viridis" palette be used. Default <code>FALSE</code> .
col_ebird	Logical, Alternative colour palette for trend values, based on the colour palette used for eBird status and trend maps.
strata_custom	(sf) Data Frame. Data frame of modified existing stratification, or a <code>sf</code> spatial data frame with polygons defining the custom stratifications. See details on <code>strata_custom</code> in <code>stratify()</code> .
zoom_range	Logical. When <code>TRUE</code> (default) zooms into region of the map where trend data are available. If <code>FALSE</code> , map extends out to cover all of the stratification map. Zoom-in uses <code>ggplot2::coord_sf()</code>

Value

a `ggplot2` plot

See Also

Other indices and trends functions: [generate_indices\(\)](#), [generate_trends\(\)](#), [plot_geofacet\(\)](#), [plot_indices\(\)](#)

Examples

```
# Using the example model for Pacific Wrens...

# Generate the continental and stratum indices
i <- generate_indices(pacific_wren_model)

# Now generate trends
t <- generate_trends(i, slope = TRUE)

# Generate the map (without slope trends)
plot_map(t)

# Generate the map (with slope trends)
plot_map(t, slope = TRUE)

# Viridis
plot_map(t, col_viridis = TRUE)

# Generate a map (with alternate column - lower 95% Credible limit)
plot_map(t, alternate_column = "trend_q_0.05")
```

prepare_data	<i>Filter for data quality</i>
--------------	--------------------------------

Description

Check and filter the stratified data by minimum required samples for modelling, and prepare data format for use by models.

Usage

```
prepare_data(
  strata_data,
  min_year = NULL,
  max_year = NULL,
  min_n_routes = 3,
  min_max_route_years = 3,
  min_mean_route_years = 1,
  min_span = 2,
  quiet = FALSE,
  assume_observer_variation_log_normal = FALSE
)
```

Arguments

strata_data	List. Stratified data generated by <code>stratify()</code>
min_year	Numeric. Minimum year to use. Default (NULL) uses first year in data.

<code>max_year</code>	Numeric. Maximum year to use. Default (NULL) uses first year in data.
<code>min_n_routes</code>	Numeric. Required minimum routes per strata where species has been observed. Default 3.
<code>min_max_route_years</code>	Numeric. Required minimum number of years with non-zero observations of species on at least 1 route. Default 3. Only retain strata with at least one route where the species was observed at least once in this many years.
<code>min_mean_route_years</code>	Numeric. Required minimum average of years per route with the species observed. Default 1. Only retain strata where the average number of years the species was observed per route is greater than this value.
<code>min_span</code>	Numeric. Required minimum span of years with surveys conducted in a stratum. Default 2. Only retain strata where the time-span between the earliest and most recent surveys is greater than this value. Must be less than the number of years between <code>min_year</code> and <code>max_year</code> . Useful for removing strata that only have observations in recent years when running long-term analyses. For example many northern strata in BCRs 1 - 3 (Alaska, Northwest Territories, and Nunavut) have relatively new BBS routes that have only been surveyed since the early 2000s. Long-term analyses that include these strata can generate extremely uncertain population trajectories in the years with no data.
<code>quiet</code>	Logical. Suppress progress messages? Default FALSE.
<code>assume_observer_variation_log_normal</code>	Logical. Default FALSE. If FALSE, the model will generate indices that adjust only for temporal variation in observers and so generate annual indices that are the expected counts in a given year averaged across all observers and all routes in the stratum. FALSE option may be useful in situations where one or more of the 3 assumptions below are questionable, including where the observer variation is not log-normal (e.g., heavy-tailed, or skewed). The difference can be important because it can change the relative scaling of annual indices among strata, which in turn influences the weight of each strata-level trend on trends for composite regions. If TRUE, the annual indices adjust for both spatial and temporal variation in observers by using the same <code>retrans_obs</code> retransformation factor for all strata, and so generate annual indices that are the expected counts in a given year averaged across all routes in the stratum and assume all observer variation is 1) noise, 2) approximately log-normally distributed ($\text{retrans_obs} = 0.5 \cdot \text{sdobs}^2$), and observers are exchangeable. This alternate TRUE is a more appropriate estimate of the uncertainty in an expected count, generating a prediction error across all observers in the analysis, however even small departures from the assumption of log-normally distributed errors can introduce bias into the annual indices, relative to the observed counts in the dataset.

Value

List of prepared (meta) data to be used for modelling and further steps.

- `model_data` - list of data formatted for use in Stan modelling
- `meta_data` - meta data defining the analysis
- `meta_strata` - data frame listing strata meta data
- `raw_data` - data frame of summarized counts used to create `model_data` (just formatted more nicely)

See Also

Other Data prep functions: [prepare_model\(\)](#), [prepare_spatial\(\)](#), [stratify\(\)](#)

Examples

```
# Toy example with Pacific Wren sample data

# First, stratify the sample data

s <- stratify(by = "bbs_cws", sample_data = TRUE,
             use_map = FALSE)

# Prepare the stratified data for use in a model. In this
# toy example, we will set the minimum year as 2009 and
# maximum year as 2018, effectively only setting up to
# model 10 years of data.

p <- prepare_data(s, min_year = 2009, max_year = 2018)
```

<code>prepare_model</code>	<i>Prepare model parameters</i>
----------------------------	---------------------------------

Description

Calculate and format the prepared data for use in modelling. Different parameters are defined for different types of models (see `?bbs_models` for a list of models included in `bbsBayes2`).

Usage

```
prepare_model(
  prepared_data,
  model,
  model_variant = "hier",
  model_file = NULL,
  use_pois = FALSE,
  heavy_tailed = TRUE,
  n_knots = NULL,
  basis = "mgcv",
  calculate_nu = FALSE,
  calculate_log_lik = FALSE,
```

```

    calculate_cv = FALSE,
    cv_k = 10,
    cv_fold_groups = "obs_n",
    cv_omit_singles = TRUE,
    use_likelihood = TRUE,
    predict_counts = TRUE,
    set_seed = NULL,
    quiet = FALSE
  )

```

Arguments

<code>prepared_data</code>	List. Prepared data generated by <code>prepare_data()</code> (if <code>model_variant</code> is not <code>spatial</code>) or <code>prepare_spatial()</code> (if <code>model_variant</code> is <code>"spatial"</code>).
<code>model</code>	Character. Type of model to use, must be one of <code>"first_diff"</code> (First Differences), <code>"gam"</code> (General Additive Model), <code>"gamye"</code> (General Additive Model with Year Effect), or <code>"slope"</code> (Slope model).
<code>model_variant</code>	Character. Model variant to use, must be one of <code>"nonhier"</code> (Non-hierarchical), <code>"hier"</code> (Hierarchical; default), or <code>"spatial"</code> (Spatially explicit).
<code>model_file</code>	Character. Optional location of a custom Stan model file to use.
<code>use_pois</code>	Logical. Whether to use an Over-Dispersed Poisson model (<code>TRUE</code>) or an Negative Binomial model (<code>FALSE</code> ; default).
<code>heavy_tailed</code>	Logical. Whether extra-Poisson error distributions should be modelled as a t-distribution, with heavier tails than the standard normal distribution. Default <code>TRUE</code> . Recent results suggest this is best even though it requires much longer convergence times. Can only be set to <code>FALSE</code> with Poisson models (i.e. <code>use_pois = TRUE</code>).
<code>n_knots</code>	Numeric. Number of knots for <code>"gam"</code> and <code>"gamye"</code> models
<code>basis</code>	Character. Basis function to use for GAM smooth, one of <code>"original"</code> or <code>"mgcv"</code> . Default is <code>"original"</code> , the same basis used in Smith and Edwards 2020. <code>"mgcv"</code> is an alternate that uses the <code>"tp"</code> basis from the package <code>mgcv</code> (also used in <code>brms</code> , and <code>rstanarm</code>). If using the <code>"mgcv"</code> option, the user may want to consider adjusting the prior distributions for the parameters and their precision.
<code>calculate_nu</code>	Logical. Whether to calculate the <code>nu</code> parameter as a factor of <code>gamma(2, 0.1)</code> . Default <code>FALSE</code> .
<code>calculate_log_lik</code>	Logical. Whether to calculate point-wise log-likelihood of the data given the model. Default <code>FALSE</code> .
<code>calculate_cv</code>	Logical. Whether to use <code>bbsBayes2</code> ' cross validation. Note this is experimental . See Details.
<code>cv_k</code>	Numeric. The number of K folds to include (only relevant if <code>calculate_cv = TRUE</code>). Default 10. Note this is experimental .
<code>cv_fold_groups</code>	Character. The data column to use when determining the grouping level of the observations to be assigned to different fold groups. Must be one of

	<code>obs_n</code> (default) or <code>routes</code> (only relevant if <code>calculate_cv = TRUE</code>). Note this is experimental . See the models article for more details.
<code>cv_omit_singles</code>	Logical. Whether to omit test groups with no replication (only relevant if <code>calculate_cv = TRUE</code>). Default <code>TRUE</code> . See the models article for more details.
<code>use_likelihood</code>	Logical. If set to <code>FALSE</code> generates prior predictions i.e., ignores the likelihood of the count data. Default <code>TRUE</code>
<code>predict_counts</code>	Logical. Whether to generate log-scale mean predictions for each count. Default is <code>TRUE</code> , so that each model output can be used to run posterior predictive checks, but if set to <code>FALSE</code> it reduces the size of the final stored model object.
<code>set_seed</code>	Numeric. If <code>NULL</code> (default) no seed is set. Otherwise an integer number to be used with <code>withr::with_seed()</code> internally to ensure reproducibility.
<code>quiet</code>	Logical. Suppress progress messages? Default <code>FALSE</code> .

Details

There are ways you can customize the model run. You can supply a custom `model_file` created with the `copy_model_file()` function and then edited manually.

You can edit or overwrite the initialization values (`init_values`) in the output of `prepare_model()` to customize the values supplied to `cmdstanr::sample()`. In previous versions of `bbsBayes2`, these initial values were specific to the model and model version. Since version 1.1.3, the initial values supplied by default in `prepare_model()` are a single value `$init_values = 1`, which This initializes all parameters randomly between `-1` and `1` on the unconstrained parameter space. You can change these initial values and supply alternate initialization values in any way that `cmdstanr::sample()` accepts the `init` argument, using the `init_alternate` argument in `run_model()`.

To implement `bbsBayes2`' version of cross validation, set `calculate_cv = TRUE`. You can set up your own system for cross validation by modifying the `folds` list-item in the output of `prepare_model()`. **Note this is considered experimental.**

See the [models article](#) for more advanced examples and explanations.

Value

A list of prepared data.

- `model_data` - list of data formatted for use in Stan modelling
- `init_values` - list of initialization parameters
- `folds` - a vector of k-fold groups each observation is assigned to (if `calculate_cv = TRUE`), or `NULL`
- `meta_data` - meta data defining the analysis
- `meta_strata` - data frame listing strata meta data
- `raw_data` - data frame of summarized counts used to create `model_data` (just formatted more nicely)

See Also

Other Data prep functions: [prepare_data\(\)](#), [prepare_spatial\(\)](#), [stratify\(\)](#)

Examples

```
s <- stratify(by = "bbs_cws", sample_data = TRUE,
             use_map = FALSE)
p <- prepare_data(s)
pm <- prepare_model(p, model = "first_diff", model_variant = "hier")
```

prepare_spatial	<i>Define neighbouring strata for spatial analyses</i>
-----------------	--

Description

Given the prepared data and a spatial data frame of polygons outlining strata, identify a neighbourhood matrix for use in modelling.

Usage

```
prepare_spatial(
  prepared_data,
  strata_map,
  voronoi = FALSE,
  nearest_fill = FALSE,
  island_link_dist_factor = 1.2,
  buffer_type = "buffer",
  buffer_dist = 10000,
  add_map = NULL,
  label_size = 3,
  queen = FALSE,
  quiet = FALSE
)
```

Arguments

prepared_data	List. Prepared data generated by prepare_data() .
strata_map	sf Data Frame. sf map of the strata in (MULTI)POLYGONS. Must have column "strata_name" matching strata output from prepare_data() .
voronoi	Logical. Whether or not to use Voroni method. Default FALSE .
nearest_fill	Logical. For strata with no neighbours, whether or not to fill in by centroids of the 2 nearest neighbours when not using the Voronoi method. Default FALSE .
island_link_dist_factor	Numeric. Distances within a factor of this amount are considered nearest strata neighbours. Used when linking otherwise isolated islands of strata, when not using the Voronoi method. Default 1.2.

buffer_type	Character. Which buffer type to use when using the Voronoi method. Must be one of buffer (default) or convex_hull . See Details for specifics.
buffer_dist	Numeric. Distance to buffer and link the strata if not connected when using the Voronoi method. Units are that of <code>sf::st_crs(strata_map)</code> . This is the <i>starting</i> distance if buffer_type = "buffer" or the final distance if buffer_type = "convex_hull". Default 10000. See Details.
add_map	<code>sf</code> object. Spatial data to add to map output.
label_size	Numeric. Size of the labels on the map. For data with many different strata it can be useful to reduce the size of the labels. Default 3.
queen	Logical. Default is FALSE. if TRUE, a single shared boundary point defines strata as neighbours, if FALSE, more than one shared point is required. With a regular rectangular grid, if TRUE each grid-cell will have 8 neighbours, (analogous to the queen's moves in chess). If FALSE, each grid-cell will have 4 neighbours (analogous to the rook's move). note that for irregular grids more than one shared boundary point does not necessarily mean a shared boundary line. Setting to TRUE can increase the realised spatial smoothing of parameters in the model because it increases the number of neighbours with which each strata is sharing information. Value is passed to <code>spdep::poly2nb()</code>
quiet	Logical. Suppress progress messages? Default FALSE.

Details

When using the Voronoi method, a buffer is used to fill around and link strata together. If the **buffer_type** is **buffer**, **buffer_dist** is the starting distance over which to buffer. If not all strata are linked, this distance is increased by 10% and applied again, repeating until all strata are linked. If **buffer_type** is **convex_hull**, then a convex hull is used to link up the strata before applying a buffer at a distance of **buffer_dist**. Note that all distances are in the units of `sf::st_crs(strata_map)`.

Value

List of prepared (meta) data to be used for modelling and further steps.

- **spatial_data** - list of samples, nodes, adjacent matrix and map visualizing the matrix
- **model_data** - list of data formatted for use in Stan modelling
- **meta_data** - meta data defining the analysis
- **meta_strata** - data frame listing strata meta data
- **raw_data** - data frame of summarized counts used to create **model_data** (just formatted more nicely)

See Also

Other Data prep functions: [prepare_data\(\)](#), [prepare_model\(\)](#), [stratify\(\)](#)

Examples

```
map <- load_map("bbs_cws")
s <- stratify(by = "bbs_cws", sample_data = TRUE,
              use_map = FALSE)
p <- prepare_data(s, min_max_route_years = 2)
sp <- prepare_spatial(p, map)

# Visually explore the spatial linkages
sp$spatial_data$map

# Overlay subset strata map on original mapping data
sp <- prepare_spatial(p, map, add_map = map)
sp$spatial_data$map
```

remove_cache	<i>Remove bbsBayes2 cache</i>
--------------	-------------------------------

Description

Remove all or some of the data downloaded via `fetch_bbs_data()` as well as model executables created by `cmdstanr::cmdstan_model()` via `run_model()`.

Usage

```
remove_cache(type = "bbs_data", level = "all", release = "all")
```

Arguments

type	Character. Which cached data to remove. One of "all", "bbs_data", or "models". If "all", removes entire cache directory (and all data contained therein). If "bbs_data", removes only BBS data downloaded with <code>fetch_bbs_data()</code> . If "models", removes only model executables compiled when <code>run_models()</code> is run.
level	Character. BBS data to remove, one of "all", "state", or "stop". Only applies if <code>type = "bbs_data"</code> . Default "all".
release	Character/Numeric. BBS data to remove, one of "all", 2020, or years 2022 through 2026. Only applies if <code>type = "bbs_data"</code> . Default "all".

Value

Nothing

See Also

Other BBS data functions: [fetch_bbs_data\(\)](#), [have_bbs_data\(\)](#), [load_bbs_data\(\)](#)

Examples

```
## Not run:
# Remove everything
remove_cache(type = "all")

# Remove all BBS data files (but not the dir)
remove_cache(level = "all", release = "all")

# Remove all 'stop' data
remove_cache(level = "stop", release = "all")

# Remove all 2020 data
remove_cache(level = "all", release = 2020)

# Remove 2020 stop data
remove_cache(level = "stop", release = 2020)

# Remove all model executables
remove_cache(type = "model")

## End(Not run)
```

run_model

Run Bayesian model

Description

Run Bayesian model with `cmdstandr::sample()` using prepare data and model parameters specified in previous steps.

Usage

```
run_model(
  model_data,
  refresh = 100,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 1000,
  iter_sampling = 1000,
  adapt_delta = 0.8,
  max_treedepth = 11,
  k = NULL,
  output_basename = NULL,
  output_dir = ".",
  save_model = TRUE,
  overwrite = FALSE,
  retain_csv = FALSE,
```

```

    set_seed = NULL,
    quiet = FALSE,
    show_exceptions = FALSE,
    init_alterate = NULL,
    compiler_optimization = "O1",
    ...
)

```

Arguments

model_data	List. Model data generated by <code>prepare_model()</code> .
refresh	Numeric. Passed to <code>cmdstanr::sample()</code> . Number of iterations between screen updates. If 0, only errors are shown.
chains	Numeric. Passed to <code>cmdstanr::sample()</code> . Number of Markov chains to run.
parallel_chains	Numeric. Passed to <code>cmdstanr::sample()</code> . Maximum number of chains to run in parallel.
iter_warmup	Numeric. Passed to <code>cmdstanr::sample()</code> . Number of warmup iterations per chain.
iter_sampling	Numeric. Passed to <code>cmdstanr::sample()</code> . Number of sampling (post-warmup) iterations per chain.
adapt_delta	Numeric. Passed to <code>cmdstanr::sample()</code> . The adaptation target acceptance statistic.
max_treedepth	Numeric. Passed to <code>cmdstanr::sample()</code> . The maximum allowed tree depth for the NUTS engine. See <code>?cmdstanr::sample</code> .
k	Numeric. The K-fold group to run for cross-validation. Only relevant if folds defined by <code>prepare_model(calculate_cv = TRUE)</code> or custom definition. See <code>?prepare_model</code> or the models article for more details.
output_basename	Character. Name of the files created as part of the Stan model run and the final model output RDS file if <code>save_model = TRUE</code> . Defaults to a character string that is unique to the species, model, model_variant, and system time of <code>model_run()</code> call (nearest minute).
output_dir	Character. Directory in which all model files will be created. Defaults to the working directory, but recommend that the user sets this to a particular existing directory for better file organization.
save_model	Logical. Whether or not to save the model output to file as an RDS object with all required data. Defaults to <code>TRUE</code> .
overwrite	Logical. Whether to overwrite an existing model output file when saving.
retain_csv	Logical. Whether to retain the Stan csv files after the model has finished running and the fitted object has been saved. Defaults to <code>FALSE</code> because csv files duplicate information saved in the model output file save object, when <code>save_model = TRUE</code> , and so for file organization and efficient use of memory, these are deleted by default.

<code>set_seed</code>	Numeric. If NULL (default) no seed is set. Otherwise an integer number to be used with <code>withr::with_seed()</code> internally to ensure reproducibility.
<code>quiet</code>	Logical. Suppress progress messages? Default FALSE.
<code>show_exceptions</code>	Logical. Passed to <code>cmdstanr::sample()</code> . Defaults to FALSE. When TRUE, prints all informational messages from Stan, for example rejection of the current proposal. Disabled by default in <code>bbsBayes2</code> , because of the copious informational messages during the initialization period that have no bearing on model fit. If fitting a custom model, recommend setting this to TRUE.
<code>init_alterate</code>	Passed to <code>init</code> argument in <code>cmdstanr::sample()</code> . Replaces the initial values in the <code>model_data[["init_values"]]</code> created by <code>prepare_model</code> . Should accept any of the acceptable approaches to setting <code>inits</code> argument in <code>?cmdstanr::sample</code> .
<code>compiler_optimization</code>	Optional character indicator to pass to compiler <code>cmdstanr::cmdstan_model(stanc_options =)</code> . Default is "O1" (the basic optimization), common options include "O", "O1", "Oexperimental" or NULL (the default for <code>cmdstanr</code>). Depending on the model, these alternate optimization options may improve or worsen sampling efficiency. If the model has been compiled previously you will need to manually remove the previously compiled model file. (i.e., if you modify this argument and then you see the message "Model executable is up to date!", then you must remove the previously compiled model). To remove previously compiled models delete the model file from the package directory <code>bbsBayes2::bbs_dir()</code> .
...	Other arguments passed on to <code>cmdstanr::sample()</code> .

Details

The model is set up in `prepare_model()`. The `run_model()` function does the final (and often long-running) step of actually running the model. Here is where you can tweak how the model will be run (iterations etc.).

See the [models article](#) for more advanced examples and explanations.

Value

List model fit and other (meta) data.

- `model_fit` - cmdstanr model output
- `model_data` - list of data formatted for use in Stan modelling
- `meta_data` - meta data defining the analysis
- `meta_strata` - data frame listing strata meta data
- `raw_data` - data frame of summarized counts

See Also

Other modelling functions: [copy_model_file\(\)](#), [save_model_run\(\)](#)

Examples

```
s <- stratify(by = "bbs_cws", sample_data = TRUE,
             use_map = FALSE)
p <- prepare_data(s)
pm <- prepare_model(p, model = "first_diff", model_variant = "hier")

# Run model (quick and dirty)
m <- run_model(pm, iter_warmup = 20, iter_sampling = 20, chains = 2)
```

save_model_run	<i>Save output of run_model()</i>
----------------	-----------------------------------

Description

This function closely imitates `cmdstanr::save_object()` but saves the entire model output object from `run_model()` which contains more details regarding data preparation (stratification etc.).

Usage

```
save_model_run(
  model_output,
  retain_csv = TRUE,
  path = NULL,
  quiet = FALSE,
  save_file_path = NULL
)
```

Arguments

model_output	List. Model output generated by <code>run_model()</code> .
retain_csv	Logical Should the Stan csv files be retained. Defaults to TRUE if user calls function directly. However, when this function is called internally by <code>run_model</code> this is set to FALSE.
path	Character. Optional file path to use for saved data. Defaults to the file path used for the original run.
quiet	Logical. Suppress progress messages? Default FALSE.
save_file_path	Character. file path and full file name, including .rds extension.

Details

Files are saved to **path**, or if not provided, to the original location of the Stan model run files (if the original files exist).

Value

Nothing. Creates an `.rds` file at `path`.

See Also

Other modelling functions: `copy_model_file()`, `run_model()`

Examples

```
# By default, the model is saved as an RDS file during `run_model()`

# But you can also deliberately save the file (here with an example model)
save_model_run(pacific_wren_model, path = "my_model.rds")

# Clean up
unlink("my_model.rds")
```

search_species	<i>Search for species</i>
----------------	---------------------------

Description

A helper function for finding the appropriate species name for use in `stratify()`.

Usage

```
search_species(species, combine_species_forms = TRUE)
```

Arguments

species	Character/Numeric. Search term, either name in English or French, AOU code, or scientific genus or species. Matches by regular expression but ignores case.
combine_species_forms	Logical. Whether or not to search the combined species data or the uncombined species. Note that this results in different species names.

Value

Subset of the BBS species data frame matching the species pattern.

See Also

Other helper functions: `assign_prov_state()`, `load_map()`

Examples

```
# Search for various terms
search_species("Paridae")
search_species("chickadee")
search_species("mésang")
search_species("Poecile")
search_species(7360)
search_species(73)
search_species("^73") # Use regex to match aou codes starting with 73
search_species("blue grouse")
search_species("sooty grouse")

# To combine or not
search_species("blue grouse", combine_species_forms = FALSE)
search_species("sooty grouse", combine_species_forms = FALSE)
search_species("northern flicker")
search_species("northern flicker", combine_species_forms = FALSE)
```

species_forms

Species forms

Description

Species forms which will be combined if `combine_species_forms` is TRUE in `stratify()`.

Usage

```
species_forms
```

Format

species_forms:

A data frame with 13 rows and 5 columns

- `aou_unid` - The AOU id number which will identify the combined unidentified form
- `english_original` - The English name of the original 'unidentified' form
- `english_combined` - The English name of the new 'combined' forms
- `french_combined` - The French name of the new 'combined' forms
- `aou_id` - The AOU id numbers of all the forms which will be combined

Examples

```
species_forms
```

<code>species_notes</code>	<i>Species notes</i>
----------------------------	----------------------

Description

Introduced in version 1.1.2. List of species and comments returned as warning messages if the time-span of an analysis is not appropriate: A collection of species and species-complexes where observations are not consistently recorded through time. List intended to help flag particular species where some trend analyses may be inappropriate.

Usage

```
species_notes
```

Format

`species_notes`:

A data frame with 6 rows and 5 columns

- `english` - The English name of the species
- `french` - Le nom francais, de l'espece
- `aou` - The unique numerical identification of the species in the BBS database
- `minimum_year` - The suggested first year by which the species observations are consistent
- `warning` - The text warning and explanation delivered by the `prepare_data` function

Examples

```
species_notes
```

<code>stratify</code>	<i>Stratify and filter Breeding Bird Survey data</i>
-----------------------	--

Description

Assign count data to strata and filter by species of interest. Routes are assigned to strata based on their geographic location and the stratification specified by the user. Species are filtered by matching English, French or Scientific names to those in the BBS species data (see `search_species()` for a flexible search to identify correct species names).

Usage

```
stratify(
  by,
  species,
  strata_custom = NULL,
  combine_species_forms = TRUE,
  release = 2026,
  sample_data = FALSE,
  return_omitted = FALSE,
  quiet = FALSE,
  use_map = TRUE,
  distance_to_strata = NULL
)
```

Arguments

by	Character. Stratification type. Either an established type, one of "bbs", "prov_state", "bcr", "latlong", "bbs_cws", "bbs_usgs", or a custom name (see strata_custom for details).
species	Character. Bird species of interest. Can be specified by English, French, or Scientific names, or AOU code. Use search_species() for loose matching to find the exact name/code needed.
strata_custom	(sf) Data Frame. Data frame of modified existing stratification, or a sf spatial data frame with polygons defining the custom stratifications. See Details.
combine_species_forms	Logical. Whether to combine ambiguous species forms. Default TRUE . See Details.
release	Numeric. Which yearly release to use, 2022 (including data through 2021 field season) or 2020 (including data through 2019). Default 2022.
sample_data	Logical. Use sample data (just Pacific Wrens). Default FALSE .
return_omitted	Logical. Whether or not to return a data frame of route-years which were omitted during stratification as they did not overlap with any stratum. For checking and troubleshooting. Default FALSE .
quiet	Logical. Suppress progress messages? Default FALSE .
use_map	Logical. Whether to stratify the BBS counts and routes based on the spatial location of route starting points relative to the polygons in the strata map. If one of the bcr_old , prov_state , bbs_cws , latlong , or bbs_usgs is supplied to the by argument, the FALSE option allows for the routes to be stratified based on the route-specific regional allocation in the BBS database. Ignored if the user supplies a custom stratification, or the bcr or bbs stratifications are supplied to the by argument. Default, TRUE
distance_to_strata	numerical. Maximum distance (in meters), within which routes with starting locations that do not intersect any strata will be joined to the

nearest stratum. This is `NULL` by default, indicating that BBS routes with starting locations that do not overlap a stratum will be omitted. If supplied, any route with a starting location that falls outside of the supplied strata polygons, but less than `distance_to_strata` meters from the boundary of at least one of the strata polygons, will be treated as if they overlap the nearest stratum. This argument is particularly useful to include routes with starting locations on or near the coast or shoreline of large lakes, which may be otherwise excluded due to plotting errors. For example, the map associated with the standard `bbs` stratification excludes 3,877 surveys on 72 routes when this argument is `NULL`. All of these 72 routes have starting locations on the coasts. Setting this argument to 4000 (any route within 4 km of a polygon) ensures all of these coastal routes are included. Default `NULL`. Caution: Users should be cautious of using this argument if the strata map does not represent the full landmass of Canada and the United States. For example using a subset of an existing strata map, such as all of the `bbs` strata within one country will link routes within the set distance of the national border (e.g., some routes in the US will be joined to strata in Canada) the same way it treats routes that happen to fall just off the coast of the supplied strata map.

Details

To define a custom subset of an existing stratification, specify the stratification in `by` (e.g., `"bbs_cws"`) and then supply a subset of `bbs_strata[["bbs_cws"]]` to `strata_custom` (see examples).

To define a completely new custom stratification, specify the name you would like use in `by` (e.g., `"east_west_divide"`) and then supply a spatial data frame with polygons identifying the different strata to `strata_custom`. Note that this data must have a column called exactly `strata_name` which names all the strata contained (see examples). This column must also be of class character not numeric or factor.

If `combine_species_forms` is `TRUE` (default), species with multiple forms (e.g., `"unid. Dusky Grouse / Sooty Grouse"`) are included in overall species groupings (i.e., `"unid."` are combined with `"Dusky Grouse"` and `"Sooty Grouse"` into `"Blue Grouse (Dusky/Sooty)"`). If the user wishes to keep the forms separate, `combine_species_forms` can be set to `FALSE`. See the data frame `species_forms`, for which species are set to be combined with which other species.

See `vignette("stratification", package = "bbsBayes2")` and the article [custom stratification](#) for more details.

Value

List of (meta) data.

- `meta_data` - meta data defining the analysis
- `meta_strata` - data frame listing strata names and area for all strata relevant to the data (i.e. some may have been removed due to lack of count data). Contains at least `strata_name` (the label of the stratum), and `area_sq_km` (area of the stratum).
- `birds_strata` - data frame of stratified count-level data filtered by species

- `routes_strata` - data frame of stratified route-level data filtered by species

See Also

Other Data prep functions: [prepare_data\(\)](#), [prepare_model\(\)](#), [prepare_spatial\(\)](#)

Examples

```
# Sample Data - BBS strata updated (2025) BBS strata with new BCRs
s <- stratify(by = "bbs", sample_data = TRUE)
# omits some routes so using distance_to_strata to capture coastal routes
s <- stratify(by = "bbs", sample_data = TRUE, distance_to_strata = 2000)

# Full data - species and stratification
# Use `search_species()` to get correct species name

# Stratify by CWS BBS strata
s <- stratify(by = "bbs_cws", species = "Common Loon",
             use_map = FALSE)

# Use use English, French, Scientific, or AOU codes for species names
s <- stratify(by = "bbs_cws", species = "Plongeon huard",
             use_map = FALSE)
s <- stratify(by = "bbs_cws", species = 70,
             use_map = FALSE)
s <- stratify(by = "bbs_cws", species = "Gavia immer",
             use_map = FALSE)

# Stratify by updated (2025) Bird Conservation Regions
s <- stratify(by = "bcr", species = "Great Horned Owl",
             distance_to_strata = 4000)

# Stratify by former CWS BBS strata
s <- stratify(by = "bbs_cws", species = "Canada Jay",
             use_map = FALSE)

# Stratify by State/Province/Territory only
s <- stratify(by = "prov_state", species = "Common Loon",
             use_map = FALSE)

# Stratify by blocks of 1 degree of latitude X 1 degree of longitude
s <- stratify(by = "latlong", species = "Snowy Owl",
             use_map = FALSE)

# Check routes omitted by stratification
s <- stratify(by = "latlong", species = "Snowy Owl", return_omitted = TRUE)
s[["routes_omitted"]]

# Use combined or non-combined species forms

search_species("Sooty grouse")
```

```

s <- stratify(by = "bbs_usgs", species = "Blue Grouse (Dusky/Sooty)",
              use_map = FALSE)
nrow(s$birds_strata) # Contains all Dusky, Sooty and unidentified

search_species("Sooty grouse", combine_species_forms = FALSE)
s <- stratify(by = "bbs_usgs", species = "unid. Dusky Grouse / Sooty Grouse",
              combine_species_forms = FALSE,
              use_map = FALSE)
nrow(s$birds_strata) # Contains *only* unidentified

# Stratify by a subset of an existing stratification
library(dplyr)
my_cws <- filter(bbs_strata[["bbs_cws"]], country_code == "CA")
s <- stratify(by = "bbs_cws", strata_custom = my_cws, species = "Snowy Owl",
              use_map = FALSE)

my_bcr <- filter(bbs_strata[["bcr_old"]], strata_name == "BCR8")
s <- stratify(by = "bcr_old", strata_custom = my_bcr,
              species = "Yellow-rumped Warbler (all forms)",
              use_map = FALSE)

# Stratify by Custom stratification, using sf map object
# e.g. with WBPBS stratum boundaries 2019
# available: https://ecos.fws.gov/ServCat/Reference/Profile/142628

## Not run:
map <- sf::read_sf("../WBPBS_Stratum_Boundaries_2019") %>%
  rename(strata_name = STRAT) # stratify expects this column

s <- stratify(by = "WBPBS_2019", strata_map = map)

## End(Not run)

```

Index

- * **BBS data functions**
 - fetch_bbs_data, 9
 - have_bbs_data, 20
 - load_bbs_data, 22
 - remove_cache, 37
- * **Data prep functions**
 - prepare_data, 30
 - prepare_model, 32
 - prepare_spatial, 35
 - stratify, 44
- * **datasets**
 - bbs_data_sample, 4
 - bbs_models, 5
 - bbs_strata, 6
 - equal_area_crs, 8
 - pacific_wren_model, 24
 - species_forms, 43
 - species_notes, 44
- * **helper functions**
 - assign_prov_state, 3
 - load_map, 23
 - search_species, 42
- * **indices and trends functions**
 - generate_indices, 10
 - generate_trends, 14
 - plot_geofacet, 24
 - plot_indices, 26
 - plot_map, 28
- * **model assessment functions**
 - get_convergence, 18
 - get_model_vars, 19
 - get_summary, 19
- * **modelling functions**
 - copy_model_file, 7
 - run_model, 38
 - save_model_run, 41

assign_prov_state, 3

assign_prov_state(), 23, 42

bbs_data_sample, 4

bbs_models, 5

bbs_strata, 6

bbsBayes2-defunct, 6, 7

bbsBayes2-deprecated, 7, 7

copy_model_file, 7

copy_model_file(), 40, 42

equal_area_crs, 8

fetch_bbs_data, 9

fetch_bbs_data(), 21, 22, 37

generate_indices, 10

generate_indices(), 17, 25, 28, 29

generate_trends, 14

generate_trends(), 13, 25, 28, 29

get_convergence, 18

get_convergence(), 19, 20

get_model_vars, 19

get_model_vars(), 18, 20

get_summary, 19

get_summary(), 18, 19

have_bbs_data, 20

have_bbs_data(), 10, 22, 37

have_cmdstan, 21

load_bbs_data, 22

load_bbs_data(), 10, 21, 37

load_map, 23

load_map(), 4, 42

pacific_wren_model, 24

plot_geofacet, 24

plot_geofacet(), 13, 17, 28, 29

plot_indices, 26

plot_indices(), 13, 17, 25, 29

plot_map, 28

plot_map(), 13, 17, 25, 28

prepare_data, 30
prepare_data(), 35, 36, 47
prepare_model, 32
prepare_model(), 32, 36, 47
prepare_spatial, 35
prepare_spatial(), 32, 35, 47

remove_cache, 37
remove_cache(), 10, 21, 22
run_model, 38
run_model(), 8, 42

save_model_run, 41
save_model_run(), 8, 40
search_species, 42
search_species(), 4, 23
species_forms, 43
species_notes, 44
stratify, 44
stratify(), 32, 35, 36