Using roccupy with data from auk

```
library(roccupy)
library(auk)
#> auk 0.4.2 is designed for EBD files downloaded after 2019-08-15.
#> No EBD data directory set, see ?auk_set_ebd_path to set EBD_PATH
#> eBird taxonomy version:  2019
```

The goal of this vignette is to illustrate how data can be converted to the format used in `roccupy`. In particular, we'll be looking at how to convert data from the `auk` package to the required format. Although we will be using only a very small portion of eBird data – the amount that is included with `auk` – the steps should be the same for much larger analyses using eBird.

First, we'll follow the steps from the `auk` introduction to obtain a small sample of zero-filled records from eBird. If you'd like to know more about what these steps are doing, please read the introduction linked.

```
# to produce zero-filled data, provide an EBD and sampling event data file
f_ebd <- system.file("extdata/zerofill-ex_ebd.txt", package = "auk")
f_smp <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk")
filters <- auk_ebd(f_ebd, file_sampling = f_smp) %>%
  auk_complete()
filters
#> Input
#>   EBD: /home/martin/R/x86_64-pc-linux-gnu-library/4.0/auk/extdata/zerofill-ex_ebd.txt
#>   Sampling events: /home/martin/R/x86_64-pc-linux-gnu-library/4.0/auk/extdata/zerofill-ex_sampling.t
#>
#> Output
#>   Filters not executed
#>
#> Filters
#>   Species: all
#>   Countries: all
#>   States: all
#>   Counties: all
#>   BCRs: all
#>   Bounding box: full extent
#>   Years: all
#>   Date: all
#>   Start time: all
#>   Last edited date: all
#>   Protocol: all
#>   Project code: all
#>   Duration: all
#>   Distance travelled: all
#>   Records with breeding codes only: no
#>   Complete checklists only: yes
```

```
ebd_sed_filtered <- auk_filter(filters,
                               file = "ebd-filtered.txt",
                               file_sampling = "sampling-filtered.txt",
```

```
                                overwrite = TRUE)
ebd_sed_filtered
#> Input
#>   EBD: /home/martin/R/x86_64-pc-linux-gnu-library/4.0/auk/extdata/zerofill-ex_ebd.txt
#>   Sampling events: /home/martin/R/x86_64-pc-linux-gnu-library/4.0/auk/extdata/zerofill-ex_sampling.t
#>
#> Output
#>   EBD: /home/martin/projects/roccupy/vignettes/ebd-filtered.txt
#>   Sampling events: /home/martin/projects/roccupy/vignettes/sampling-filtered.txt
#>
#> Filters
#>   Species: all
#>   Countries: all
#>   States: all
#>   Counties: all
#>   BCRs: all
#>   Bounding box: full extent
#>   Years: all
#>   Date: all
#>   Start time: all
#>   Last edited date: all
#>   Protocol: all
#>   Project code: all
#>   Duration: all
#>   Distance travelled: all
#>   Records with breeding codes only: no
#>   Complete checklists only: yes

ebd_zf <- auk_zerofill(ebd_sed_filtered)
```

The zero-filled data consists of two tables. One lists the species observations:

```
str(ebd_zf$observations)
#> tibble [1,857 x 4] (S3: tbl_df/tbl/data.frame)
#>  $ checklist_id     : chr [1:1857] "G1731477" "G1731477" "G1731477" "G1733362" ...
#>  $ scientific_name  : chr [1:1857] "Alcedo meninting" "Halcyon smyrnensis" "Todiramphus chloris" "Al
#>  $ observation_count: chr [1:1857] "0" "0" "0" "0" ...
#>  $ species_observed : logi [1:1857] FALSE FALSE FALSE FALSE TRUE TRUE ...
```

The other lists the details of each checklist, such as the time spent looking:

```
str(ebd_zf$sampling_events)
#> tibble [619 x 31] (S3: tbl_df/tbl/data.frame)
#>  $ checklist_id            : chr [1:619] "S11332512" "S12570927" "S10896855" "S13581013" ...
#>  $ last_edited_date        : chr [1:619] "2017-05-27 10:25:40" "2017-07-05 13:20:48" "2017-05-27 10
#>  $ country                 : chr [1:619] "Singapore" "Singapore" "Singapore" "Singapore" ...
#>  $ country_code            : chr [1:619] "SG" "SG" "SG" "SG" ...
#>  $ state                   : chr [1:619] "Singapore" "Singapore" "Singapore" "Singapore" ...
#>  $ state_code              : chr [1:619] "SG-" "SG-" "SG-" "SG-" ...
#>  $ county                  : chr [1:619] NA NA NA NA ...
#>  $ county_code             : chr [1:619] NA NA NA NA ...
#>  $ iba_code                : chr [1:619] NA NA NA NA ...
#>  $ bcr_code                : int [1:619] NA NA NA NA NA NA NA NA NA NA ...
#>  $ usfws_code              : chr [1:619] NA NA NA NA ...
#>  $ atlas_block             : chr [1:619] NA NA NA NA ...
```

```
#>  $ locality                 : chr [1:619] "Singapore - Esplanade" "Singapore Botanic Gardens" "Chang
#>  $ locality_id              : chr [1:619] "L1654642" "L952084" "L1337067" "L2090485" ...
#>  $ locality_type            : chr [1:619] "P" "H" "H" "P" ...
#>  $ latitude                 : num [1:619] 1.29 1.31 1.36 1.29 1.4 ...
#>  $ longitude                : num [1:619] 104 104 104 104 104 ...
#>  $ observation_date         : Date[1:619], format: "2012-08-12" "2012-11-29" ...
#>  $ time_observations_started: chr [1:619] "09:30:00" "18:15:00" "13:00:00" "09:00:00" ...
#>  $ observer_id              : chr [1:619] "obs160139" "obs135493" "obs149858" "obs286526" ...
#>  $ sampling_event_identifier: chr [1:619] "S11332512" "S12570927" "S10896855" "S13581013" ...
#>  $ protocol_type            : chr [1:619] "Traveling" "Incidental" "Stationary" "Traveling" ...
#>  $ protocol_code            : chr [1:619] "P22" "P20" "P21" "P22" ...
#>  $ project_code             : chr [1:619] "EBIRD" "EBIRD_CAN" "EBIRD" "EBIRD" ...
#>  $ duration_minutes         : int [1:619] 90 NA 25 60 60 65 120 120 90 50 ...
#>  $ effort_distance_km       : num [1:619] 1.61 NA NA 3.22 NA ...
#>  $ effort_area_ha           : num [1:619] NA NA NA NA NA NA NA NA NA NA ...
#>  $ number_observers         : int [1:619] 1 2 1 1 1 2 1 1 1 2 ...
#>  $ all_species_reported     : logi [1:619] TRUE TRUE TRUE TRUE TRUE TRUE ...
#>  $ group_identifier         : chr [1:619] NA NA NA NA ...
#>  $ trip_comments            : chr [1:619] NA "with P.Jantunen. Just a quick dash from the airport be
#>  - attr(*, ".internal.selfref")=<externalptr>
#>  - attr(*, "unique")= logi TRUE
```

First, we'll reshape the observations into the format required for `roccupy`. What we need is a DataFrame of shape `(n_checklists, n_species)`, rather than the long format produced by `auk`. Luckily, we can easily switch to the required format using the `reshape2` package:

```
library(reshape2)

y_checklist <- dcast(ebd_zf$observations, checklist_id ~ scientific_name,
                     value.var = 'species_observed')

species_names <- unique(ebd_zf$observations$scientific_name)

str(y_checklist)
#> 'data.frame':    619 obs. of  4 variables:
#>  $ checklist_id       : chr  "G1731477" "G1733362" "G2469935" "G2469955" ...
#>  $ Alcedo meninting   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
#>  $ Halcyon smyrnensis : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
#>  $ Todiramphus chloris: logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
```

In the following, we will have to drop some records since they are missing information. To make this a bit easier, we'll put together the observations and checklist details into a single DataFrame:

```
combined <- merge(ebd_zf$sampling_events, y_checklist, by='checklist_id')
```

Now, we need to find some environmental covariates and define our sites. Here, we'll use the `worldclim` variables to do both: the covariates in `worldclim` will be our environmental covariates, and the cells in the raster will be our sites. We emphasise here that this example is *not* meant to be ecologically meaningful: the amount of data is small and it is collected in a very small area, so it is unlikely that the inferences are very insightful.

```
library(raster)
#> Loading required package: sp

r <- getData("worldclim",var="bio",res=2.5)
```

Let's fetch the cell numbers, which will be our sites:

```
cell_nums <- cellFromXY(r, cbind(combined$longitude, combined$latitude))
table(cell_nums)
#> cell_nums
#> 18366809 18366810 18366811 18366813 18366814 18366815 18375450 18375451
#>        3       36        6        4        1        5        3        2
#> 18375452 18375453 18375454 18375455 18375456 18384089 18384090 18384091
#>        6        8        7       15       28      291        9       19
#> 18384092 18384093 18384094 18384095 18384096 18384097 18384098 18392729
#>       35        6        1        1        8        6        1        4
#> 18392730 18392731 18392732 18392733 18392734 18401372 18401373 18401376
#>        1        2       45       10        2        9       19        5
#> 18410012 18410013 18418649 18418651
#>       10        6        1        4
```

We have some repeat observations, which is good. To keep track of the cell numbers, add them to the sampling information:

```
combined$env_cell_num <- cell_nums
```

Now we'll fetch the environmental covariates in these cells:

```
unique_cells <- unique(cell_nums)
cell_vals <- raster::extract(r, unique_cells)

X_env <- data.frame(cell_vals)
```

Some of them are NA. We will drop these. We will also drop checklists without durations.

```
X_env$cell_num <- unique_cells
na_cells <- rowSums(is.na(X_env)) > 0
to_drop <- unique_cells[na_cells]

# Drop NA cells from sampling info:
rel_combined <- combined[!(combined$env_cell_num %in% to_drop), ]

# Also only keep those with durations recorded
rel_combined <- rel_combined[!is.na(rel_combined$duration_minutes), ]

# Drop the NA environmental cells
rel_X_env <- X_env[!(X_env$cell_num %in% to_drop), ]
```

The next step is key: we need to match up the checklists with the corresponding sites. We can use R's `match` function to make this easy. However, crucially, sites are numbered from *zero* onwards, so we need to subtract 1 from the result of `match`:

```
# Now, we need to encode the cell ids
cell_ids <- match(rel_combined$env_cell_num, rel_X_env$cell_num)

# IMPORTANT: These need to start from zero! So:
cell_ids <- cell_ids - 1
```

Now we're pretty much good to go! Let's fit an example model with three environmental covariates and log-transformed duration:

```
env_vars_to_use <- c('bio1', 'bio2', 'bio4')
```

```
# Scale the environmental variables:
to_scale <- rel_X_env[, env_vars_to_use]
scaled_X_env <- scale(to_scale)
scaled_X_env_df <- data.frame(scaled_X_env)

env_formula <- paste(env_vars_to_use, collapse='+')

env_formula
#> [1] "bio1+bio2+bio4"
```

```
X_obs <- data.frame(log_duration = log(rel_combined$duration_minutes))

obs_formula <- 'log_duration'
```

Now we have all we need and can fit the model:

```
checklist_cell_ids <- cell_ids

y_checklist <- rel_combined[, species_names]

fit_result <- msod_vi(env_formula, obs_formula, scaled_X_env_df, X_obs,
                      y_checklist, checklist_cell_ids)
```

This object can now be used in the usual way. For example, we can take a look at the draws for the observation part of the model:

```
coef_draws <- coef(fit_result)
lapply(coef_draws$obs_slopes, summary)
#> $`Alcedo meninting`
#>    Intercept        log_duration
#>  Min.   :-4.7737   Min.   :-0.65181
#>  1st Qu.:-2.9178   1st Qu.:-0.16174
#>  Median :-2.8027   Median :-0.06398
#>  Mean   :-2.8094   Mean   :-0.06426
#>  3rd Qu.:-2.6830   3rd Qu.: 0.04055
#>  Max.   :-0.9432   Max.   : 0.35021
#>
#> $`Halcyon smyrnensis`
#>    Intercept       log_duration
#>  Min.   :-3.571   Min.   :0.1029
#>  1st Qu.:-2.841   1st Qu.:0.2039
#>  Median :-2.743   Median :0.2343
#>  Mean   :-2.740   Mean   :0.2339
#>  3rd Qu.:-2.643   3rd Qu.:0.2640
#>  Max.   :-1.466   Max.   :0.3767
#>
#> $`Todiramphus chloris`
#>    Intercept        log_duration
#>  Min.   :-3.9081   Min.   :0.3413
#>  1st Qu.:-2.8619   1st Qu.:0.4580
#>  Median :-2.7650   Median :0.4885
#>  Mean   :-2.7569   Mean   :0.4910
#>  3rd Qu.:-2.6548   3rd Qu.:0.5229
#>  Max.   :-0.1688   Max.   :0.7033
```

Please see the `ebird-example` vignette for more detail about how to interpret the results of the model.