

# Appendix S7 - Extracting Migration Status Information

*Nicholas J Clark, David J Harris, Ceridwen Fraser*

This appendix describes how to extract information on species' migratory status from BirdLife International. This data can be accessed in the `BBS.occurrences` library using `data('Bird.mig.status')`. Note, the primary functions in this appendix were adapted from code that was generously provided by [Tad Dallas](#).

Some libraries that are not automatically included in the package will be needed for web-scraping. Check that they are installed and up-to-date

```
library(tidyverse)
library(httr)
library(rvest)
library(purrr)
library(stringr)
library(parallel)
```

Download the Birdtree.org taxonomy file, which will be necessary for searching common names when species' binomials are not recognised by BirdLife. Note, this step requires an internet connection

```
temp <- tempfile()
download.file("https://data.vertlife.org/birdtree/BLIOCPhyloMasterTax.csv",
             temp)
BT.tax <- read.csv(temp)
```

Load the binary species occurrence data that is included in the `BBS.occurrences` package and extract the column names

```
library(BBS.occurrences)
data("BBS.occurrences")
sp.names <- colnames(BBS.occurrences)
```

Here we perform a sanity check to ensure that parallel loading is supported on our machine. Note, the library `parallel` needs to be loaded for this to function properly. If parallel computation is not desired, just set `n_cores` to 1

```
n_cores <- 3

if (n_cores > 1) {
  cl <- makePSOCKcluster(n_cores)
  setDefaultCluster(cl)

  # Check errors when loading on each cluster
  test_load1 <- try(clusterEvalQ(cl, library(rvest)),
    silent = TRUE)

  # If errors, iterate over options for loading
  if (class(test_load1) == "try-error") {

    # Try finding paths using system.file()
    pkgLibs <- unique(c(sub("/rvest$", "", system.file(package = "rvest"))))
    clusterExport(NULL, c("pkgLibs"), envir = environment())
    clusterEvalQ(cl, .libPaths(pkgLibs))

    # Check again for errors
    test_load2 <- try(clusterEvalQ(cl, library(rvest)),
      silent = TRUE)

    if (class(test_load2) == "try-error") {

      # Try loading .libPath() directly
      clusterEvalQ(cl, .libPaths(as.character(.libPaths()))))
      test_load3 <- try(clusterEvalQ(cl, library(rvest)),
        silent = TRUE)

      if (class(test_load3) == "try-error") {

        parallel_compliant <- FALSE
        stopCluster(cl)

      } else {
        parallel_compliant <- TRUE
      }

    } else {
      parallel_compliant <- TRUE
    }

  } else {
    parallel_compliant <- TRUE
  }
} else {
  parallel_compliant <- FALSE
}

parallel_compliant
```

After confirming parallel capabilities, start the parallel clusters and run the function. Note, this step requires an internet connection and will take some time to process the 303 avian species in the dataset

```
# Export necessary data and variables to each
# cluster
clusterExport(NULL, c("BT.tax", "sp.names"), envir = environment())

# Export necessary libraries
clusterEvalQ(cl, library(rvest))
clusterEvalQ(cl, library(stringr))
clusterEvalQ(cl, library(purrr))
clusterEvalQ(cl, library(xml2))

search_migstatus <- pbapply::pblapply(seq_along(sp.names),
  function(x) {

    session <- rvest::html_session("http://datazone.birdlife.org/species/search")
    Sys.sleep(4)

    form <- html_form(session)[[3]]

    filledform <- set_values(form, kw = sp.names[x])
    session <- submit_form(session, filledform)$url

    landing_pg <- xml2::read_html(session)

    # Find text that forms the species' hyperlink
    sp_link <- rvest::html_nodes(landing_pg, css = "tr") %>%
      rvest::html_attrs() %>% purrr::flatten_chr() %>%
      purrr::keep(~grepl("rowClick", .x)) %>%
      stringr::str_replace("rowClick\\(\\'\\'",
        "'") %>% stringr::str_replace("\\'\\'",
        "'")

    # If binomial not found, try the common name
    if (identical(sp_link, character(0))) {
      session <- rvest::html_session("http://datazone.birdlife.org/species/search")
      Sys.sleep(4)

      comm_name <- as.character(BT.tax$English[which(BT.tax$TipLabel ==
        paste(sp.names[x]))])
      form <- rvest::html_form(session)[[3]]

      filledform <- rvest::set_values(form, kw = comm_name)
      session <- rvest::submit_form(session,
        filledform)$url
      landing_pg <- xml2::read_html(session)

      sp_link <- rvest::html_nodes(landing_pg,
        css = "tr") %>% rvest::html_attrs() %>%
        purrr::flatten_chr() %>% purrr::keep(~grepl("rowClick",
          .x)) %>% stringr::str_replace("rowClick\\(\\'\\'",
            "'") %>% stringr::str_replace("\\'\\'",
            "'")
    }
  })
```

```

# If still not found, return NA
if (identical(sp_link, character(0))) {
  output <- data.frame(Species = sp.names[x],
    Migrate.status = "NA")
} else {
  details_link <- paste0("http://datazone.birdlife.org/species/factsheet/",
    sp_link[1], "/details/")

  migrate_status <- xml2::read_html(details_link) %>%
    rvest::html_nodes(css = "tr:nth-child(1) td:nth-child(2)") %>%
    rvest::html_text()
  output <- data.frame(Species = sp.names[x],
    Migrate.status = migrate_status)
}
} else {
  details_link <- paste0("http://datazone.birdlife.org/species/factsheet/",
    sp_link[1], "/details/")

  migrate_status <- xml2::read_html(details_link) %>%
    rvest::html_nodes(css = "tr:nth-child(1) td:nth-child(2)") %>%
    rvest::html_text()
  output <- data.frame(Species = sp.names[x],
    Migrate.status = migrate_status)
}
}
output
}, cl = cl)
Bird.mig.status <- do.call(rbind, search_migstatus)
Bird.mig.status$Migrate.status <- as.character(Bird.mig.status$Migrate.status)

```

Save the migration information in the analysis folder. This dataframe can be accessed in the BBS.occurrences library using `data('Bird.mig.status')`.

```

save(Bird.mig.status, file = "./Analysis_data/
  Bird.mig.status.Rdata")

```