Project FeederWatch: Zero-filling and taxonomic roll-up

Wesley Hochachka

Last compiled on August 03, 2022

Contents

Purpose

Provide R code that enables users working with Project FeederWatch data to perform key data manipulations that are relevant to most valid uses of the data:

1. Zero-filling (i.e., adding counts of zero birds for species that were not reported as being detected during an observation period)
2. Taxonomic roll-up (i.e., combining subspecies and distinct taxonomic forms when multiple species codes exist for focal taxa)

How to use this document

To use the R code in this document, you'll need to download the most up-to-date version of R and RStudio, and have some knowledge of the programming language R.

To run the example code, you'll need to download the sample checklist data from the Project FeederWatch website or the GitHub repository.

We also reference a "species translation table," which isn't needed to run the example code, but is important for determining which species code one needs. This table can be found on the Project FeederWatch website.

If you would like the most recent version of the complete FeederWatch dataset, it is also available on the FeederWatch website. Note that the FeederWatch data are updated annually on or about June 1. The historic versions are available in a Mendeley Data archive.

Introduction

Project FeederWatch is a place-based program that asks participants to identify and count the birds that visit backyards, nature centers, community areas, and other locales in North America from November-April. Historically, the program has focused on sites that provide supplementary feeding stations (i.e., bird feeders), but in the 2021-2022 season the project invited people to participate even if they didn't have a feeder. Moving forward participants are only required to have an area with plantings, habitat, water, or food that attracts birds.

At this time, the Project FeederWatch team recommends researchers exclude sites that do not provide supplementary feeding stations until we can verify the best way to work with this new data.

The Project FeederWatch data is publicly available and distributed as .csv files compressed into ZIP archives. Those interested in using the data are strongly encouraged to consult the Project FeederWatch data report for tips on proper use of the data (Bonter and Greig 2021).

For most uses, researchers will want to manipulate the data in one or two of the following important ways before the data can be used for meaningful analyses:

1. **Zero-filling:** Information on when birds were not reported as being detected should be added to the downloaded data (when appropriate) by inserting records with counts of zero birds.
2. **Taxonomic Roll-up:** The identities of some bird species may need to be modified because some of the reported information is being stored as subspecies or other recognizable forms (e.g., "Oregon" Junco instead of Dark-eyed Junco).

Data structure

There are two data sets that make up the Project FeederWatch data: (1) checklists (i.e., bird counts) and (2) site descriptions. This document focuses on the checklist data, and does not go into detail on each data field

in these data sets, the data collection protocols, or the data validation processes. To learn more, see the Data Dictionary available on the Project FeederWatch site and the published data report (Bonter and Greig 2021).

The Project FeederWatch checklist data available for download do not contain counts of zero, only observations of presence. More specifically, each row in the checklist data features either the maximum number of each bird species seen in their count site over a 2-day checklist period or a "nobird" entry to indicate that no birds were seen on that 2-day checklist period. However, because the FeederWatch protocol instructs participants to record all species seen within the count area, counts of zero birds can be inferred and added to the available data.

What does "zero-filling" mean and why is it important?

Zero-filling is adding counts of zero birds for species that were not detected during an observation period. These zeros indicate either that a species was absent, or that the species was present but the participant did not notice it. These two alternatives need to be kept in mind when analyzing zero-filled data.

For analyses, counts of zero are extremely important. For example, week-to-week changes in whether any birds of a species are seen at a site can provide information about birds' motivation or need to obtain food from bird feeders. As another example, both the non-zero counts and the zero counts are needed to accurately describe changes in birds' distribution and numbers through time. Without knowing which locations have counted zero birds of a species, there is no way of knowing whether the apparent geographic distribution of a species is a description of the actual distribution or merely the distribution of observers. Additionally, changes in numbers through time are a product both of changes in the numbers of individuals at a site at which the species was reported, as well as changes in the proportion of sites at which at least at least one individual was seen. Both the non-zero counts and the zero counts are needed to accurately describe changes in numbers through time. See Guillera-Arroita et al. (2015) for a more detailed discussion of the importance of zero-count information.

As noted above, the double meaning of a zero—either as a true absence of a species, or merely the absence of detection of the species—does need to be taken into account. There are two approaches that can be used to correctly interpret the zero counts. One approach is to remove the ambiguity between true absence and lack of detection by using statistical methods such as occupancy models (see Zuckerberg et al. 2010 for an example using data from Project FeederWatch) or N-mixture models (Goldstein et al. 2022). Because these two methods require data to be collected on repeated visits of the same set of locations, FeederWatch data are well suited for these methods.

The second approach to correct interpretation of zero counts is to treat the zeros as indicators of "relative occurrence". This means one must assume that some constant proportion of the zero counts were at places and times at which the species was actually present but was undetected. The key is to analyze data in a way that tries to ensure that each zero has the same likelihood of meaning that "the species was present but was not detected". This is done by analyzing the data in Generalized Linear Models in which predictors of variation in the probability of detection have been included so that this variation is statistically removed.

With Project FeederWatch the major source of variation in detection rate of a species that we have identified is the amount of time that an observer watches their site during an observation period. This amount of observation effort is described by two variables in the FeederWatch data: the number of half-days of observation within the two-day observation period, and the estimated number of effort-hours across the entire two-day period. We have found that both of these variables should be included as predictors whenever possible; however, only the first of these was recorded in the earliest years of Project FeederWatch. See States et al. (2009) for an example of how we have done this in our own analysis.

Important considerations when zero-filling

The process of zero-filling, as described here, is appropriate in any observational data, including Project FeederWatch data, under two conditions:

1. We have knowledge of observation periods during which the species of interest was not seen.

2. It is reasonable to expect that the species of interest would have been reported if it had been detected.

For more information, refer to the section, "How to decide if zero-filling is appropriate?" in this document where we have summarized the issues that one needs to consider when wanting to zero-fill Project FeederWatch data for their species of interest.

What does "taxonomic roll-up" mean and why is it important?

A taxonomic roll-up is combining subspecies and distinct taxonomic forms when multiple species codes exist for focal taxa.

Project FeederWatch participants record the maximum numbers of individual birds that they have seen at the same time during a two-day checklist period. For a subset of species, participants are able to specify the level of recognizable form instead of just reporting their observations at the taxonomic level of species.

These recognizable forms are stored in the Cornell Lab of Ornithology database using a "species code" that follows eBird's taxonomic system, which is updated annually. We have created a species translation table following this taxonomy that provides the species-level code for each recognizable form, as well as the Latin, English, Canadian French, and Mexican Spanish names for each species.

Examples of species with multiple recognizable forms include Dark-eyed Junco and Fox Sparrow. It is possible for an observer to report some individual birds as coming from a recognizable form and other individuals from the species without identifying their form, and this can occur even during the same two-day checklist period. Species like Rock Pigeon also have more than one value of SPECIES_CODE: (1) the generic species, (2) truly wild Rock Pigeons inside their native range (these are not found in data from Project FeederWatch), and (3) feral Rock Pigeons. While most records of Rock Pigeons in the Project FeederWatch data are recorded under the generic species code of "rocpig", a non-trivial proportion of records specify the feral form using the species code "rocpig1".

Entering information on recognizable forms is optional in Project FeederWatch. To enter a recognizable form, a participant must enter information not provided on their standard "species to expect" list. As a result, one cannot assume that the lack of a recognizable form at a site means that the recognizable form was not present. The probability of a report of a recognizable form is some unknowable joint probability of (1) a recognizable form being present at a feeder site and (2) the probability that an observer will be motivated enough to report that bird as a recognizable form instead of as a member of a full species. This second probability, participant motivation, will almost certainly vary among participants, across the continent, and through time (because the ease of reporting recognizable forms has increased through time).

Thus, under most circumstances, if one wants to work with all of the data from such species, they will need to make sure that all data from a species will have the same species code value. Following the jargon used for this same data manipulation when working with data from eBird using the R package `auk`, we are referring to this process as a "taxonomic roll-up."

Important considerations when engaging in taxonomic roll-up

We have created code for a taxonomic roll-up that is relatively simple and assumes Project FeederWatch participants are not double-counting the same individual birds both as belonging to a recognizable form and as members of the full species.

Double-counting (the erroneous inflation of the number of individual birds being reported for a species as a whole) could occur if a participant wanted to record that they had seen some unusual individual from a recognizable form, but then also assume they needed to combine that unusual individual with the count of the rest of the individual birds seen of that species.

Hypothetical example: A participant who normally only sees Dark-eyed Juncos of the "Slate-colored" form notices an "Oregon" form one day. The participant could report that "Oregon" junco as one observation record, and additionally group the single unusual junco in with all of the other juncos, reporting them all

under the the species code for the full species, "daejun." There may be ways of assessing whether this is happening frequently, for example, by testing whether unexpectedly high counts of individual birds are reported when both a recognizable form and the full species are reported during a single observation period.

The process of performing a taxonomic roll-up as described below could be modified such that counts are only added together if all individuals are reported to be of a recognizable form and there are no individuals recorded as coming from the full species. An alternative approach could be to take the maximum number of individuals reported for any of the recognizable forms and include that value as the number observed for the species.

Individual users of the data will need to decide how they want to handle the data from recognizable forms. All users of Project FeederWatch data need to think about this process, and include in the Methods section of any manuscript their decisions and reasoning. Please refer to the section, "Additional complexities with taxonomic roll-up" for a summary of issues one needs to consider about taxonomic roll-up.

An R function for zero-filling and taxonomic roll-up

In this section we show the R code for creating a function that will both zero-fill data for one or more species, and perform a taxonomic roll-up. The function has been set-up so that the default behavior will be to pass the data through the taxonomic roll-up process before the zero-filling process. However, depending on what is most appropriate, the user has the ability to turn off the taxonomic roll-up process.

At the end of this section we present examples of using this function. If you want to understand how this function works, there is a detailed explanation of how we use R for the zero-filling and taxonomic roll-up processes in later sections of this document.

Initial set-up in R

Load the `tidyverse` package. Also set the working directory to be where your copy of the data files are located.

```
# Load the tidyverse package
library(tidyverse)

# Assign directory if not already done
# MyDirectory <- "D:/Project_FeederWatch_Analysis"
# Set working directory (i.e., where R looks for the data and stores output)
# setwd(MyDirectory)
```

Read in the sample checklist data.

```
PFW2021 <- read_csv("PFW_R_data_sample.zip")
```

```
## Rows: 2943003 Columns: 28
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (11): LOC_ID, SUBNATIONAL1_CODE, ENTRY_TECHNIQUE, SUB_ID, OBS_ID, PROJ_P...
## dbl (16): LATITUDE, LONGITUDE, Month, Day, Year, HOW_MANY, VALID, REVIEWED, ...
## lgl  (1): PLUS_CODE
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The generalized zero-filling and taxonomic roll-up function

Creating a function is, in the simplest cases, nothing more than bundling together a set of R code so that all of the code can be run by simply typing the name that is given to the function. The user will add

"arguments" inside parentheses immediately after the name of the function. Arguments to a function provide a convenient way for users to specify changes to a small number of aspects of how the function works, for example specifying the table of data on which the function needs to act.

The function, named `ZeroFillPFW`, is defined using the command `function()`and accepts three arguments:

1. `InputData`, the name of the data table (can be data.frame or a tibble) that will be used for zero-filling,
2. `SpeciesCodes`, a vector of one or more SPECIES_CODE values for which zero-filling will be performed, and
3. `rollup`, a logical (TRUE/FALSE) indicator of whether a taxonomic roll-up should be done with the default behavior being to pass the data through the roll-up process before the zero-filling step.

Everything enclosed within the curly brackets is the set of R commands that will be run when the function is run.

```
ZeroFillPFW <- function(InputData, SpeciesCodes, rollup = TRUE){

  #The first three steps collect information that
  # will be necessary for taxonomic roll-up and zero-filling. First, we create
  # a table that contains only the information describing the observation
  # periods (and not anything specific to any single birds species). Second,
  # we need to count the number of bird species for which zero-filling has been
  # requested, in order to know how many times we will need to loop through the
  # zero-filling process. Third, we need to check the species codes for which
  # zero-filling was requested by comparing these to a list of all species codes
  # found in InputData.

  #Create the table of sampling event information from all observation periods
  PFW_SED <- InputData %>%
    select(-OBS_ID, -SPECIES_CODE, -alt_full_spp_code, -HOW_MANY, -PLUS_CODE,
           -VALID, -REVIEWED) %>%
    distinct()

  #Count the number of species codes
  NSpecies <- length(SpeciesCodes)

  #Identify any species codes that are not found in the data
  InvalidSppCodes <- InputData %>%
    select(SPECIES_CODE) %>%
    distinct() %>%
    anti_join(y = ., x = tibble(SPECIES_CODE = SpeciesCodes), by = "SPECIES_CODE")
  #For each non-existing species code, print an error message
  if (nrow(InvalidSppCodes) >= 1){
    for (i in 1:nrow(InvalidSppCodes)){
      cat(paste("THE SPECIES CODE \'",
                InvalidSppCodes$SPECIES_CODE[i],
                "\' DOES NOT EXIST\n",
                sep = ""))
      cat(" IN THE DATA TABLE, AND ZERO-FILLING IS NOT POSSIBLE.\n\n")
    }
    #break out of the function by returning a null result
    cat("THIS FUNCTION IS STOPPING BECAUSE ZERO-FILLING OF ALL\n")
    cat(" REQUESTED SPECIES CODES IS NOT POSSIBLE.\n\n")
    return(NULL)
  }
```

```
#Loop through the species codes in turn, possibly doing a taxonomic roll-up,
# and always zero-filling each species' data.
for (i in 1:NSpecies){
  #
  #Check if this species code is "nobird" and if so skip over this
  # species code.
  if (SpeciesCodes[i] == "nobird"){
    cat("THE SPECIES CODE \'nobird\' INDICATES AN ABSENCE OF BIRDS,\n")
    cat(" AND IS NOT AN ACTUAL BIRD SPECIES. NO ZERO-FILLING WILL\n")
    cat("BE DONE FOR THE SPECIES_CODE \'nobird\'.\n\n")
    next
  }
  #Place the SPECIES_CODE for the ith species into Spp_to_Zero_Fill.
  Spp_to_Zero_Fill <- SpeciesCodes[i]
  #
  #If a taxonomic roll-up is to be done, then do so for the focal species.
  if (rollup){
    InputData <- InputData %>%
      mutate(SPECIES_CODE = case_when(
        (is.na(alt_full_spp_code)
         ~ SPECIES_CODE),
        ((!is.na(alt_full_spp_code) &
            alt_full_spp_code == Spp_to_Zero_Fill)
         ~ alt_full_spp_code),
        ((!is.na(alt_full_spp_code) &
            alt_full_spp_code != Spp_to_Zero_Fill)
         ~ SPECIES_CODE))
      )
  }

  #Create template for zero-count records for the focal species.
  FakeSppSpecificFields <- as_tibble_row(list(OBS_ID = "OBSnull",
                                              SPECIES_CODE = Spp_to_Zero_Fill,
                                              HOW_MANY = 0,
                                              PLUS_CODE = NA,
                                              VALID = 1,
                                              REVIEWED = 0))

  #Zero-fill the data for the focal species.
  ZeroFilledFocalSpecies <- InputData %>%
    filter(SPECIES_CODE == Spp_to_Zero_Fill) %>%
    bind_rows(., bind_cols(PFW_SED, FakeSppSpecificFields)) %>%
    group_by(across(c(names(PFW_SED), "SPECIES_CODE"))) %>%
    summarize(OBS_ID = min(OBS_ID), HOW_MANY = sum(HOW_MANY),
              PLUS_CODE = max(PLUS_CODE),
              VALID = min(VALID), REVIEWED = max(REVIEWED)) %>%
    ungroup()
  ZeroFilledFocalSpecies$PLUS_CODE <- as.logical(ZeroFilledFocalSpecies$PLUS_CODE)

  #Put the zero-filled data into a table that accumulates data from all of
  # the species named in the vector SpeciesCodes. Either create the tibble
  # to be returned if it does not already exist, or append rows to an
  # existing object.
```

```
    if ((exists("OutputData")))
      OutputData <- bind_rows(OutputData, ZeroFilledFocalSpecies)
    else
      OutputData <- ZeroFilledFocalSpecies
  }
  #After all species' data are zero-filled, return the zero-filled data to the
  # user. Unless the user of the function directs the output into some object,
  # the output will be printed to the console and not be saved for subsequent use.
  #Also, if a taxonomic roll-up has been done, print a message to remind users
  # this was done.
  if (rollup){
    cat("\nNOTE: ANY RECORDS OF BIRDS REPORTED TO THE LEVEL OF A SUBSPECIES OR\n")
    cat(" OTHER RECOGNIZABLE FORM HAVE BEEN TREATED AS MEMBERS OF THE FULL\n")
    cat(" SPECIES FOR THE PURPOSE OF ZERO-FILLING, AND 'species_code' VALUES\n")
    cat(" FOR THESE RECOGNIZABLE FORMS HAVE BEEN REPLACE BY THE CODES FOR THE\n")
    cat(" FULL SPECIES IN THE OUTPUT.\n\n")
  }
  return(OutputData)
}
```

In addition to zero-filling and taxonomic roll-up code, the function is made robust to possible errors:

- The function checks whether all of the SPECIES_CODE values actually exist in the data to be zero-filled; if not, then the function will not return any output.
- There is a check that "nobird" is not listed as a species for which zero-filled data are being requested. "nobird" is not a species of bird, but instead is an indicator that no bird species was reported during a specific 2-day checklist period (i.e. for a specific SUB_ID). The "nobird" records exist in the downloaded data as a way of adding a zero-count record even when there were no bird species seen during the two-day checklist period. Otherwise, zero-count records erroneously would not exist for observation periods during which no bird species were recorded.
- A message is printed out whenever the "rollup" argument is set to "TRUE", and a taxonomic roll-up might have been done prior to zero-filling.

Examples of using the zero-filling and taxonomic roll-up function

Once the function has been defined by running the part of the R script that is inside the box immediately above, you'll be able to use the function to run through the examples below with sample checklist data from 2021.

Example 1: Two-species taxonomic roll-up and zero-filling

Use the function `ZeroFillPFW` to perform a taxonomic roll-up and zero-fill data for two species. Note that performing the taxonomic roll-up is the default and does not need to be specified.

The species referenced in this example are Rock Pigeon (rocpig) and Mourning Dove (moudov). The species codes for these and all species can be found in the species translation table mentioned at the beginning of this document and available for download from the Project FeederWatch website.

```
MyZeroFilledData <- ZeroFillPFW(InputData = PFW2021,
                                SpeciesCodes = c("rocpig", "moudov"))
```

Example 2: Single-species zero-filling without taxonomic roll-up

The purpose of this example is to show how to tell the function not to do a taxonomic roll-up. For the species used in this example, House Sparrow (species code: houspa), does not have any separate species code values

for subspecies or other recognizable forms, so that the same results will be produced regardless of whether the taxonomic roll-up is turned off or left on.

```
MyZeroFilledData_2 <- ZeroFillPFW(InputData = PFW2021,
                                  SpeciesCodes = "houspa",
                                  rollup = FALSE)
```

Example 3: Potentially misleading zero-filling without a taxonomic roll-up

The purpose of this example is to compare the results of zero-filling with and without taxonomic roll-up for a species that does have recognizable forms: the Dark-eyed Junco (species code: daejun)

We first zero-fill with a taxonomic roll-up and then zero-fill a second time without the taxonomic roll-up.

Finally, we count the number of non-zero records. The number of non-zero records without the roll-up erroneously understates the number of checklist periods during which our example species, Dark-eyed Junco (species code: daejun), was observed. This happens because none of the records using species codes from recognizable forms are treated as Dark-eyed Juncos without the taxonomic roll-up.

```
MyZeroFilledData_daejun_with_rollup <- ZeroFillPFW(InputData = PFW2021,
                                                   SpeciesCodes = "daejun",
                                                   rollup = TRUE)
```

```
## `summarise()` has grouped output by 'LOC_ID', 'LATITUDE', 'LONGITUDE',
## 'SUBNATIONAL1_CODE', 'ENTRY_TECHNIQUE', 'SUB_ID', 'Month', 'Day', 'Year',
## 'PROJ_PERIOD_ID', 'DAY1_AM', 'DAY1_PM', 'DAY2_AM', 'DAY2_PM',
## 'EFFORT_HRS_ATLEAST', 'SNOW_DEP_ATLEAST', 'Data_Entry_Method', 'user_id',
## 'observer_id', 'housing_density', 'no_birds'. You can override using the
## `.groups` argument.
```

```
##
## NOTE: ANY RECORDS OF BIRDS REPORTED TO THE LEVEL OF A SUBSPECIES OR
##  OTHER RECOGNIZABLE FORM HAVE BEEN TREATED AS MEMBERS OF THE FULL
##  SPECIES FOR THE PURPOSE OF ZERO-FILLING, AND 'species_code' VALUES
##  FOR THESE RECOGNIZABLE FORMS HAVE BEEN REPLACE BY THE CODES FOR THE
##  FULL SPECIES IN THE OUTPUT.
```

```
MyZeroFilledData_daejun_without_rollup <- ZeroFillPFW(InputData = PFW2021,
                                                      SpeciesCodes = "daejun",
                                                      rollup = FALSE)
```

```
## `summarise()` has grouped output by 'LOC_ID', 'LATITUDE', 'LONGITUDE',
## 'SUBNATIONAL1_CODE', 'ENTRY_TECHNIQUE', 'SUB_ID', 'Month', 'Day', 'Year',
## 'PROJ_PERIOD_ID', 'DAY1_AM', 'DAY1_PM', 'DAY2_AM', 'DAY2_PM',
## 'EFFORT_HRS_ATLEAST', 'SNOW_DEP_ATLEAST', 'Data_Entry_Method', 'user_id',
## 'observer_id', 'housing_density', 'no_birds'. You can override using the
## `.groups` argument.
```

```
#Print to screen the counts on numbers of observation periods
# with and without taxonomic roll-up.
MyZeroFilledData_daejun_with_rollup %>%
  filter(HOW_MANY > 0) %>%
  summarise(N_NonzeroRecords_with_rollup = n())
```

```
## # A tibble: 1 x 1
##   N_NonzeroRecords_with_rollup
##                          <int>
## 1                       165035
```

```
MyZeroFilledData_daejun_without_rollup %>%
  filter(HOW_MANY > 0) %>%
  summarise(N_NonzeroRecords_no_rollup = n())
```

```
## # A tibble: 1 x 1
##   N_NonzeroRecords_no_rollup
##                        <int>
## 1                     161181
```

Example 4: Taxonomic roll-up to above the species level

For some purposes, it can be appropriate to combine information from multiple species together before zero-filling the data. One general case for doing this is for pairs or groups of species that cannot easily be distinguished from each other, for example Black-capped and Carolina Chickadees in their zones of geographic overlap and hybridization.

Another example, considered here, are the hawks in the genus *Accipiter*. Accipiters are not easily identifiable to species, and in the FeederWatch data some sightings are even reported as being of an "Accipiter species" (species code 'accipi') instead of being identified to species. If one wanted to examine patterns of geographic or temporal change in perceived risk of depredation at sites, for instance, then it may be reasonable to combine all records of any type of *Accipiter* into a single taxonomic unit prior to zero-filling these data. This type of higher-than-species-level taxonomic roll-up will need to be done prior to running the zero-filling function.

The first step in this process is to identify all of the possible values of SPECIES_CODE that should be combined. This information can be found in the species-translation table. Searching this table for the text string "accipiter" shows that there are 8 SPECIES_CODE values for Accipiters (as of March 2022):

- **shshaw**: the generic species code for Sharp-shinned Hawks
- **shshaw2**: the code for the northern (i.e. North American) form of Sharp-shinned Hawk
- **coohaw**: the generic species code for Cooper's Hawks
- **y00612**: this species code indicates that the observer saw either a Sharp-shinned or a Cooper's Hawk, but that the observer was not comfortable in identifying the bird to the level of a species
- **norgos**: the generic species code for Northern Goshawk
- **norgos2**: the code for the North American form of Northern Goshawk
- **y00666**: this code indicates that the observer saw either a Cooper's Hawk or a Northern Goshawk, but was not comfortable in making a species-level identification
- **accipi**: the species code for "some type of *Accipiter* hawk"

We can replace all of the other species codes for *Accipiter* hawks with the code "accipi" prior to zero-filling the data for the "accipi" species code. Below is R code that specifies how to replace the species codes before using the function to zero-fill the data for this group of species.

```
PFW2021_Accipiter_rollup <- PFW2021 %>%
  mutate(SPECIES_CODE = case_when(
    (SPECIES_CODE %in% c("shshaw", "shshaw2", "coohaw", "y00612",
                         "norgos", "norgos2", "y00666")
     ~ "accipi"),
    (TRUE
     ~ SPECIES_CODE)
  ))

MyZeroFilledData_4 <- ZeroFillPFW(InputData = PFW2021_Accipiter_rollup,
                                  SpeciesCodes = c("accipi"))
```

```
## `summarise()` has grouped output by 'LOC_ID', 'LATITUDE', 'LONGITUDE',
## 'SUBNATIONAL1_CODE', 'ENTRY_TECHNIQUE', 'SUB_ID', 'Month', 'Day', 'Year',
## 'PROJ_PERIOD_ID', 'DAY1_AM', 'DAY1_PM', 'DAY2_AM', 'DAY2_PM',
```

```
## 'EFFORT_HRS_ATLEAST', 'SNOW_DEP_ATLEAST', 'Data_Entry_Method', 'user_id',
## 'observer_id', 'housing_density', 'no_birds'. You can override using the
## `.groups` argument.
##
## NOTE: ANY RECORDS OF BIRDS REPORTED TO THE LEVEL OF A SUBSPECIES OR
##  OTHER RECOGNIZABLE FORM HAVE BEEN TREATED AS MEMBERS OF THE FULL
##  SPECIES FOR THE PURPOSE OF ZERO-FILLING, AND 'species_code' VALUES
##  FOR THESE RECOGNIZABLE FORMS HAVE BEEN REPLACE BY THE CODES FOR THE
##  FULL SPECIES IN THE OUTPUT.
```

How to decide if zero-filling is appropriate?

Zero-filling is only appropriate if two conditions are met:

1. There is consistent information about the occurrence of observation periods during which the species of interest could have been seen although it was not actually seen.
2. It is reasonable to assume that the species of interest would have been reported as being seen if it had been detected.

Condition 1

The first condition is met by the Project FeederWatch data because of the nature of the project, with participants making observations from the same location multiple times throughout the project's season (November – April). There are records in the database for each 2-day checklist period (typically a 2-day weekend), even if no birds were seen.

Condition 2

The second condition may not always be valid because of the nature of how participants report observations. Because participants are presented with a list of species, there are two classes of species in the data: (1) species that are on the lists that are presented to participants and should always be reported if seen (species for which zero-filling can and should be done); and (2) write-in species that are added by participants based on their own motivations (species for which zero-filling may or may not make sense).

Participants are able to add any species that are not on the list of expected species. Because adding these species requires additional effort from the participant, it is not safe to assume that all participants have gone through the effort of adding non-typical species to their lists. This is a generalization, but not a universal rule.

For example, participants may be less inclined to report common species that are not typically seen at bird feeders, even though a feeder is not required to participate in the FeederWatch program as of the 2021-2022 season. Few participants, for instance, may be motivated enough to report species that at best could only marginally be considered to have visited a bird feeder (e.g., a Mallard on the pond beyond the bird feeders).

Conversely, many participants would be thrilled to report a truly rare, vagrant species such as a Brambling (*Fringilla montifringilla*). As of March 2022 there are eight locations that have records of Bramblings in the data of PFW. These are likely an accurate record of all of the FeederWatch locations at which this rare Eurasian vagrant has appeared.

There is no record of which species are write-ins in which regions, and the lists of species presented to participants have changed through time for various reasons (e.g., a species has been expanding its range north). Additionally, these regions have become more geographically refined throughout the project's history. Initially, the continent was simply divided into "east" and "west"; online data entry has allowed more precise lists to be generated.

How to determine is Condition 2 is met?

To determine if the second condition (reasonable to assume that the species of interest would have been reported as being seen if it had been observed, for example if a species was a write-in species), we suggest that researchers use what they know about the natural history of a species and information about how frequently a species is reported when deciding if zero-filling is appropriate.

We have provided information about how frequently a species is reported in the species translation table. This table has a column labeled "n_locations", and the numbers in this column are counts of the number of unique FeederWatch sites at which each species has been reported at least once. Species reported at many sites within their ranges are likely not write-in species and therefore good candidates for zero-filling. For ubiquitous species like the Dark-eyed Junco, "many sites" will mean tens of thousands of locations. For a range-restricted species in an area with a low density of FeederWatch participants, like Juniper Titmouse, a smaller number of locations may suffice as an indicator that the species is detected enough to warrant zero-filling.

Final considerations regarding zero-filling

We are dealing with the issue of zero filling is such detail because we believe that it is essentially always beneficial, if not essential, to zero-fill Project FeederWatch data before using them. An important reason for doing this (as already described) is that it is impossible to definitively detect changes in distribution or abundance of a species through time, unless you can distinguish between the cases that: (1) an observer was present and would have reported a species if it had been present, and (2) the absence of any data-collection activity. Only the first of these alternatives provides useful information.

We are not aware of any circumstances under which more precise and accurate results can be produced using statistical methods designed to work with "presence-only" (i.e. non-zero-filled) data. Such methods were developed for the modelling of species' distributions when zero-count information are unavailable, which is the case for records from museum collections. However, these methods are inherently limited in what they can tell us (Guillera-Arroita et al.2015). Further, these presence-only analysis methods require the creation of "background" or "pseudo-absence" locations in lieu of actual locations at which zero individuals of a species were reported. These "background" locations have no true observation event associated with them, and so they do not have any information about observer effort (i.e. the amount of time that observers spent watching their site during an observation period). As a consequence, the information about observer effort from the actual records cannot be used in an analysis, even though we know that variation in observer effort accounts for substantial amounts of the variation in the number of bird species observed and the number of individuals counted for any species (Bonter and Greig 2017). Failing to account for the effects of variable observer effort will result in lower statistical power to detect actual patterns in the data. For example, for a species distribution model, the distribution of a species will not be as clearly defined (see Figure 2 of Johnston et al. 2021).

In summary, we strongly advise zero-filling data for the species for which this is possible. If there is a species for which zero-filling is not possible and the goal is to model the distributions of that species, then we suggest reading the following papers for guidance on appropriate analytical methods: Valavi et al. (2022) and Valavi et al. (2021).

Additional complexities with taxonomic roll-up

The complexities involved in taxonomic roll-ups have been mentioned earlier in this document. As illustrated in Example 3 above, taxonomic roll-up may be essential for creating a data set that accurately describes when and where a species was reported. Conversely, the counts of the number of individual birds may be somewhat inflated following a taxonomic roll-up. In practice, anyone who is using Project FeederWatch data needs to consider the following three things about taxonomic roll-ups, and also to be able to justify in their methods the reason either for performing (or not performing) a taxonomic roll-up:

1. **A number of species will have some individual reports made to the level of a subspecies**

or other recognizable sub-group within that species. For instance, in Example 4 above, some participants have reported Sharp-shinned Hawks as belonging to the "northern" form, even though it is the only possible form within the FeederWatch region and potentially the observers did not even observe a bird closely enough to confirm without a doubt that the reported bird did indeed belong to the reported form. Basically, when using the data, always check whether the species in which you are interested has data reported from recognizable sub-specific forms. You can find whether some reports of your species of interest are at the sub-specific level by searching the species translation table.

2. The converse can also happen, with **some observations being recorded as being identified only to the level above species.** *Accipiter* hawks, and Black-capped and Carolina Chickadees are two examples of this. Always check whether your species of interest is possibly ever or often reported as a member of some species pair or larger group (species codes for pairs of species are always the letter "y" followed by 5 numbers). Very low frequencies of this are possibly ignorable noise. If you need to treat your species of interest as part of a combined group, Example 4 above provides an example of how to do so.

3. **Some participants may be entering counts of recognizable sub-specific forms both as these forms, and then again as part of the total count for the overall species.** A conservative approach would be to first check whether there are observation periods in which your species of interest is simultaneously being reported using the species code for the overall species and the species code for some recognizable form. If this occurs, the most conservative course of action would be to remove all of the data from the LOC_ID (i.e. the site) where this is found. For ubiquitous species like Dark-eyed Junco, the loss of data will be trivial. Another option would be to assess whether the total counts are detectably higher during observation periods when both sub-specific forms and overall species are simultaneously reported; after this assessment, the data in question could be retained or removed as appropriate.

Step-by-step guide: Zero-filling for one species without taxonomic roll-up:

If you are interested in seeing what is happening in the "black box" that is the zero-filling and taxonomic roll-up function, continue reading for a heavily documented example of how to zero-fill the Project FeederWatch data for one species without a taxonomic roll-up.

This zero-filling process involves three main steps:

1. Create a new row of data for every observation period, regardless of whether the species of interest was seen during that observation period.

2. Select a species to zero-fill and prepare a template for zero-count records.

3. Combine zero-count records created in step 1 and 2 with the original data. This combining is done by summing together the counts of birds seen within each separate observation period. If there is an actual observation of that species being reported, the summed count is the actual count observed (any number plus zero is the original number). If the species was not reported in an observation period, then a new record of zero birds of that species is added to the table.

In order to run the R code in this section, download the sample checklist data from the Project FeederWatch website or the GitHub repository. Then, store the data in your working directory and read the data into R.

```
PFW2021 <- read_csv("PFW_R_data_sample.zip")
```

Step 1: Create a new row of data for every 2-day checklist period

We need to create a basis for building the new records of counts of zero birds for the bird species whose data are to be zero-filled. The basis for building these zero-count records is the set of information from the actual

observations that is stored in the FeederWatch database, with each checklist period represented by a single row of information.

The R commands below create the table of sampling event data, which are the subset of variables shared across all species' records for a site-date combination. This removes the variables that provide information that is unique to a single species, and collapses these data down to a single unique copy of each record using the function `distinct()`.

```
PFW_SED <- PFW2021 %>%
  select(-OBS_ID, -SPECIES_CODE, -HOW_MANY, -PLUS_CODE,
         -VALID, -REVIEWED) %>%
  distinct()
```

Step 2: Select species to zero-fill and prepare template for zero-count records

Zero-filling needs to be done on a species-by-species basis, because each species will have a unique subset of 2-day checklist periods on which it was not observed. Once the species code for the species is assigned to an object in R, we can create a template for inserting records that zero individuals of the species were observed. This template will then be appended to each row within the sampling event data that were created above. The template contains six pieces of information that are present for every observation of a species in the original data:

1. **OBS_ID**: Each observation of a single species during a 2-day checklist period) is identified based on a unique OBS_ID value. Because we cannot assign a legitimate OBS_ID within the template, we will instead assign to the template a clearly artificial value: "OBSnull". When combining a zero-count record with an existing actual record in the second stage of zero-filling, "OBSnull" will be superseded by the existing real OBS_ID.
2. **SPECIES_CODE**: This is the code by which each taxon is referenced in the Project FeederWatch database.
3. **HOW_MANY**: This is the maximum number of individual birds observed for each species during each 2-day checklist period, and for zero-filling purposes, the appropriate number to use for the template is zero.
4. **PLUS_CODE**: The very earliest Project FeederWatch data were submitted by participants on paper "bubble" forms on which the number of birds observed for a species needed to be indicated by filling in the bubble on the form that represented this specific number. Occasionally, the actual counts of birds that were seen was greater than the value of any of the bubbles. In these cases the participant would fill in a bubble that indicated a larger-than-maximum count, and the filling in of this bubble was reported as the value "1" in the column labeled "PLUS_CODE". PLUS_CODE values are irrelevant for the zero-counts and this variable is assigned to be a missing value (represented by "NA").
5. **VALID**: When an observation of a species is being sent into the database, it passes through an automated filter that looks for unexpected species for a given geographic area, or for unexpected high counts. If a record is not unusual, it is assigned a value of "1" in the variable VALID in the database. A "0" indicates an unusual record that needs to be manually reviewed. If, upon review, the record appears to be legitimate, the value of VALID is changed to "1". Within the publicly available data, all records are available, regardless of their status based on the VALID variable. Because we want the zero-count records to remain if someone filters these data based on the VALID variable, we assign all of the new zero-count records a value of "1".
6. **REVIEWED**: All records flagged as unexpected using the VALID variable should eventually undergo a review by Project FeederWatch staff. When this review is complete, the presence of this manual review in denoted by setting the value of the variable REVIEWED to be "1"; this is done, regardless of whether the manual review led to any change in the VALID variable. The zero-count records that we create are not manually reviewed, and so receive a value of "0".

```
#Specify the species code of the focal species for zero-filling.
Spp_to_Zero_Fill <- "daejun"

#Create fake zero-count records for a species of interest.
FakeSppSpecificFields <- as_tibble_row(list(OBS_ID = "OBSnull",
                                            SPECIES_CODE = Spp_to_Zero_Fill,
                                            HOW_MANY = 0,
                                            PLUS_CODE = NA,
                                            VALID = 1,
                                            REVIEWED = 0))
```

Step 3: Do the zero-filling

Pipe the data through a series of manipulations that create the zero-filled final table of data.

The following steps are taken in this process:

1. Filter the input data down to retain only rows of input data from the focal species that have been selected for zero-filling.
2. Append to these real observations a set of fake observations in which there is a row for each observation period that is present in the input data, and in which each of these fake observations is a report of zero individuals of the focal species.
3. Group the observations based on all of the names of variables in the `PFW_SED` tibble plus the species code for the focal species. After this, all summarizing operations will act within each of the groups, with each group either composed of two rows (an actual observation event that reported the focal species and a fake record of zero individuals being reported), or just the fake records of zero individuals for observation events during which no individuals of the focal species were reported. We need to group based on all of the `PFW_SED` table's values plus SPECIES_CODE so that all of the values of these variables will appear in the output from the function `summarize()`.
4. Collapse down the pairs of real and fake records so that:
   - the OBS_ID value is the actual OBS_ID when there is a pair of records,
   - the number of birds seen is the actual value when there is a real observation, or otherwise zero,
   - the PLUS_CODE is either the real value of PLUS_CODE of otherwise it is a missing value,
   - the VALID flag is set to the true valid flag when a real observation is present, or otherwise is set to a value of "1" (i.e. a valid record), and
   - the REVIEWED flag is set to the actual value when a real observation exists, or otherwise is set to "0" (i.e. not reviewed).
5. Ungroup (remove the grouping information) so that further work with the zero-filled data will not produce weird results.
6. Set the data type of the variable PLUS_CODE back to being a Boolean ("logical") data type within R, to match the data type of the input data. Somehow the data type has been changed in the process of running through the five steps in the sequence of piped instructions.

```
ZeroFilledFocalSpecies <- PFW2021 %>%
  filter(SPECIES_CODE == Spp_to_Zero_Fill) %>%
  bind_rows(., bind_cols(PFW_SED, FakeSppSpecificFields)) %>%
  group_by(across(c(names(PFW_SED), "SPECIES_CODE"))) %>%
  summarize(OBS_ID = min(OBS_ID), HOW_MANY = sum(HOW_MANY),
                          PLUS_CODE = max(PLUS_CODE),
                VALID = min(VALID), REVIEWED = max(REVIEWED)) %>%
```

```
  ungroup() %>%
  select(LOC_ID, LATITUDE, LONGITUDE, housing_density, SUBNATIONAL1_CODE,
         ENTRY_TECHNIQUE, SUB_ID, OBS_ID, Month, Day, Year, PROJ_PERIOD_ID,
         SPECIES_CODE, HOW_MANY, PLUS_CODE, VALID, REVIEWED, DAY1_AM, DAY1_PM,
         DAY2_AM, DAY2_PM, EFFORT_HRS_ATLEAST, SNOW_DEP_ATLEAST,
         Data_Entry_Method)
ZeroFilledFocalSpecies$PLUS_CODE <- as.logical(ZeroFilledFocalSpecies$PLUS_CODE)
```

Et voila! There is now a table named `ZeroFilledFocalSpecies` that contains the zero-filled records for the focal species.

Step-by-step guide: Zero-filling for one species with taxonomy roll-up:

The process of zero-filling data is the logical place in the sequence of data processing to conduct taxonomic roll-up. This is because the roll-up process needs to take into account the possibility that multiple reports of the same species may exist for an observation event, in the case in which some individuals are reported as recognizable forms and other individuals are reported at the taxonomic level of the full species. Summing the counts of individuals within each full species and observation period is one possible approach.

Zero-filling is already summing counts of individuals from at least two rows for each species and count period when a species was present: an additional dummy record with a count of zero individual birds needs to be summed within that species and count period. This summing process, as implemented in the first example of zero-filling without taxonomic roll-up, will already automatically add together all of the two (or more) records that it finds that have the same SPECIES_CODE and SUB_ID (i.e., the zero-filling process is pre-adapted to additionally aid in the process of a taxonomic roll-up).

First, read in the data if you haven't already.

```
PFW2021 <- read_csv("PFW_R_data_sample.zip")
```

Step 1. Create a new row of data for every single observation period

Create a table that contains only the data describing the observation period (i.e., 2-day checklist) by removing all species-specific columns. Then reduce this table down to only a single record for each observation period using `distinct()`.

```
PFW_SED <- PFW2021 %>%
  select(-OBS_ID, -SPECIES_CODE, -alt_full_spp_code, -HOW_MANY, -PLUS_CODE,
         -VALID, -REVIEWED) %>%
  distinct()
```

Step 2: Replace sub-specific taxon codes for the focal species code

Roll up the records for the species whose data are to be zero-filled by replacing the species code(s) for the recognizable form with the species code for the full species. The result of this replacement is put into a new tibble so that the data read into R is free from modification in case it's needed for some other purpose later in this R session.

We use the `case_when()` function to do the replacing only for the single species for which we want to zero-fill the data. There are three possible cases we need to deal with:

1. If there is no full-species code in the field alt_full_spp_code then the current value in SPECIES_CODE is retained.

2. If there is a full-species code in the field alt_full_spp_code and that value **is** for the species in which we are interested, then the value in SPECIES_CODE is replaced by the value in alt_full_spp_code.

3. If there is a full-species code in the field alt_full_spp_code and that value is not for the species in which we are interested, then the current value in SPECIES_CODE is retained.

```
#Specify the species code of the focal species for zero-filling.
Spp_to_Zero_Fill <- "rocpig"

PFW2021_rollup <- PFW2021 %>%
  mutate(SPECIES_CODE = case_when(
    (is.na(alt_full_spp_code)
      ~ SPECIES_CODE),
    ((!is.na(alt_full_spp_code) &
      alt_full_spp_code == Spp_to_Zero_Fill)
      ~ alt_full_spp_code),
    ((!is.na(alt_full_spp_code) &
       alt_full_spp_code != Spp_to_Zero_Fill)
     ~ SPECIES_CODE))
    )
```

Step 3: Prepare template for zero-count records

```
#Create fake zero-count records for the specified focal species.
FakeSppSpecificFields <- as_tibble_row(list(OBS_ID = "OBSnull",
                                            SPECIES_CODE = Spp_to_Zero_Fill,
                                            HOW_MANY = 0,
                                            PLUS_CODE = NA,
                                            VALID = 1,
                                            REVIEWED = 0))
```

Step 4: Do the zero-filling

Pipe the data through a series of manipulations that create the zero-filled final table of data. These are the same steps as in the first example in which there was no taxonomic roll-up.

```
ZeroFilledFocalSpecies <- PFW2021_rollup %>%
  filter(SPECIES_CODE == Spp_to_Zero_Fill) %>%
  bind_rows(., bind_cols(PFW_SED, FakeSppSpecificFields)) %>%
  group_by(across(c(names(PFW_SED), "SPECIES_CODE"))) %>%
  summarize(OBS_ID = min(OBS_ID), HOW_MANY = sum(HOW_MANY),
            PLUS_CODE = max(PLUS_CODE),
            VALID = min(VALID), REVIEWED = max(REVIEWED)) %>%
  ungroup() %>%
  select(LOC_ID, LATITUDE, LONGITUDE, housing_density, SUBNATIONAL1_CODE,
         ENTRY_TECHNIQUE, SUB_ID, OBS_ID, Month, Day, Year, PROJ_PERIOD_ID,
         SPECIES_CODE, HOW_MANY, PLUS_CODE, VALID, REVIEWED, DAY1_AM, DAY1_PM,
         DAY2_AM, DAY2_PM, EFFORT_HRS_ATLEAST, SNOW_DEP_ATLEAST,
         Data_Entry_Method)
ZeroFilledFocalSpecies$PLUS_CODE <- as.logical(ZeroFilledFocalSpecies$PLUS_CODE)
```

The only difference between zero-filling with and without a taxonomic roll-up is that there is an extra step before the zero-filling in which the SPECIES_CODE values for recognizable forms are replaced by the SPECIES_CODE of the associated full species.

Replacing existing SPECIES_CODE values with new species codes that aggregate together taxa creates the

possibility of doing other sorts of taxonomic roll-ups, such as combining data from all *Accipiter* species into a single taxonomic unit by manually replacing all of the species-level or lower-level species code values with some identical above-species-level species code.

This might actually be a useful process for many uses (e.g. quantifying frequencies of predators' visits to sites) because these *Accipiter* hawk species can be challenging to identify to species. Note that in the case of *Accipiter* hawks there is already a *SPECIES_CODE* for this, "accipi", and additionally there are *species_code* values for "Sharp-shinned or Cooper's Hawk", and "Cooper's Hawk or Goshawk". All of these exist within the FeederWatch data.

References

Bonter, D.N. and E.I. Greig. 2021. Over 30 years of standardized bird counts at supplementary feeding stations in North America: A citizen science data report for Project FeederWatch. Frontiers in Ecology and Evolution 9: Article 619682 doi.org/10.3389/fevo.2021.619682

Goldstein, B. R., and P. de Valpine. 2022. Comparing N-mixture models and GLMMs for relative abundance estimation in a citizen science dataset. Scientific Reports 12:12276. doi.org/10.1038/s41598-022-16368-z

Guillera-Arroita, G., J. J. Lahoz-Monfort, J. Elith, A. Gordon, H. Kujala, P. E. Lentini, M. A. McCarthy, R. Tingley, and B. A. Wintle. 2015. Is my species distribution model fit for purpose? Matching data and models to applications. Global Ecology and Biogeography 24:276-292. doi.org/10.1111/geb.12268

Johnston, A., W. M. Hochachka, M. Stimas-Mackey, V. Ruiz-Gutierrez, O. J. Robinson, E. T. Miller, T. Auer, S. T. Kelling, and D. Fink. 2021. Analytical guidelines to increase the value of eBird data to estimate species occurrence. Diversity and Distributions 27:1265-1277. doi.org/10.1111/ddi.13271

States, S. L., W. M. Hochachka, and A. A. Dhondt. 2009. Spatial variation in an avian host community: implications for disease dynamics. Ecohealth 6:540-545. doi.org/10.1007/s10393-009-0269-2

Valavi, R., G. Guillera-Arroita, J. J. Lahoz-Monfort, and J. Elith. 2022. Predictive performance of presence-only species distribution models: a benchmark study with reproducible code. Ecological Monographs 92:e01486. doi.org/10.1002/ecm.1486

Valavi, R., J. Elith, J. J. Lahoz-Monfort, and G. Guillera-Arroita. 2021. Modelling species presence-only data with random forests. Ecography 44:1731-1742. doi.org/10.1111/ecog.05615

Zuckerberg, B., D. N. Bonter, W. M. Hochachka, W. D. Koenig, A. T. DeGaetano, and J. L. Dickinson. Climatic constraints on wintering bird distributions are modified by urbanization and weather. 2010. Journal of Animal Ecology, 80:403-413. doi.org/10.1111/j.1365-2656.2010.01780.x

Acknowledgements

Sincere thank you to the thousands of participants who are part of the Project FeederWatch community. Your dedication to recording the birds you see in your yards, and the fees you pay every season are why FeederWatch has been able to run for over 30 years.

Thank you to Dave Bonter, Emma Greig, and Rachael Mady for their feedback and revisions. Thank you as well to Christopher Sayers and Laura Vander Meiden for reviewing this documentation.

Have feedback?

This is the first documentation we have put together that includes R code and accompanying guidance for working with the FeederWatch data. Please let us know if there is anything missing or ways we could improve this documentation by contacting Project FeederWatch via email (feederwatch@cornell.edu).