Package 'PFW'

July 9, 2025

Title Filtering and Processing Data from Project FeederWatch

Version 0.1.0

Description Provides tools to import, clean, filter, and prepare Project
FeederWatch data for analysis. Includes functions for taxonomic
rollup, easy filtering, zerofilling, merging in site metadata, and more.
Project FeederWatch data comes from
<https://feederwatch.org/explore/raw-dataset-requests/>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, lubridate, httr2, xml2, stats, utils, curl, stringdist

Suggests testthat (>= 3.0.0), purrr, withr, knitr, rmarkdown

Config/testthat/edition 3

Depends R (>= 4.1.0)

URL https://github.com/ropensci/PFW, https://ropensci.github.io/PFW/

BugReports https://github.com/ropensci/PFW/issues

VignetteBuilder knitr

LazyData true

NeedsCompilation no

Author Mason W. Maron [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7170-9373>),
Sunny Tseng [rev],
Paul Carteron [rev]

Maintainer Mason W. Maron <mwmaron2@illinois.edu>

Repository CRAN

Date/Publication 2025-07-09 10:30:20 UTC

Contents

Index   **14**

---

pfw_attr   *View Filter Attributes on Manipulated Project FeederWatch Data*

Description

This function allows users to view all filters they've applied to a filtered Project FeederWatch dataset by printing its recorded filter attributes in a readable format.

Usage

```
pfw_attr(data)
```

Arguments

data   A filtered Project FeederWatch dataset.

Value

A named list of applied filters.

Examples

```
# Download/load example dataset
data <- pfw_example

# Filter for Dark-eyed Junco
filtered_data <- pfw_species(data, "Dark-eyed Junco")

# View filters applied to your active data
pfw_attr(filtered_data)
```

---

pfw_date   *Filter Project FeederWatch Data by Month and/or Year*

Description

This function filters Project FeederWatch data by year and/or month, allowing range-based filtering and wrapping months around new years.

Usage

```
pfw_date(data, year = NULL, month = NULL)
```

Arguments

data   A Project FeederWatch dataset.

year   Optional. Integer or vector of years (e.g. 2010 or 2010:2015).

month   Optional. Integer or vector of months (1–12). Supports wrapping (e.g. c(11:2) = Nov–Feb).

Value

A filtered dataset with date filter attributes.

Examples

```
# Download/load example dataset
data <- pfw_example

# Filter by a single year
data_2021 <- pfw_date(data, year = 2021)

# Filter by multiple years
data_2123 <- pfw_date(data, year = 2021:2023)

# Filter by a single month
data_feb <- pfw_date(data, month = 2)

# Filter by a span of months
data_winter <- pfw_date(data, month = 11:2)

# Filter by both year and month
data_filtered <- pfw_date(data, year = 2021:2023, month = 11:2)
```

---

pfw_dictionary   *Look Up Definitions from the Project FeederWatch Data Dictionary*

Description

This function helps users explore the FeederWatch dataset by viewing the full data dictionary or searching for definitions for specific variables.

Usage

```
pfw_dictionary(variable = NULL)
```

Arguments

variable   (Optional) A variable name (e.g., "LOC_ID") to look up. If NULL, prints the full dictionary.

Value

A printed description (for a variable) or the full dictionary.

Examples

```
# View the whole data dictionary
pfw_dictionary()

# View the data dictionary entry for location ID ("LOC_ID")
pfw_dictionary("LOC_ID")
```

---

pfw_download   *Download Raw Project FeederWatch Data by Year*

Description

This function downloads raw data for selected years from the Project FeederWatch website. It unzips the downloaded data and saves the .csv files into a local folder (default: "data-raw/"), removing the zip files afterward. It will download all files required to cover the user-selected years.

Usage

```
pfw_download(years, folder = NULL)
```

Arguments

years
: Integer or vector of years (e.g., 2001, 2001:2023, c(1997, 2001, 2023)). Data is available from 1998 to present.

folder
: The folder where Project FeederWatch data is stored. Default is "data-raw/" in a local directory.

Value

Invisibly returns the downloaded files.

Examples

```
# Download data from 2001-2006 into the default folder
pfw_download(years = 2001:2006)
```

---

pfw_example   *Example Project FeederWatch Dataset*

Description

A sample dataset for demonstration and testing purposes. This dataset includes data from 2020 - May 2024 from Washington and Oregon.

Usage

```
pfw_example
```

Format

A data frame with 556,814 rows and 24 columns.

Source

Created using `pfw_download()` and `pfw_import()` in data-raw/pfw_example.R

Examples

```
# Load the example data into the environment
data(pfw_example)

# Assign the example dataset
testing_data <- pfw_example
```

pfw_filter	*Apply Multiple Filters to Project FeederWatch Data*

Description

This function filters Project FeederWatch data by species, region, and data validity.

Usage

```
pfw_filter(
  data,
  species = NULL,
  region = NULL,
  year = NULL,
  month = NULL,
  valid = TRUE,
  reviewed = NULL,
  rollup = TRUE
)
```

Arguments

data	A Project FeederWatch dataset.
species	(Optional) A character vector of species names (common or scientific).
region	(Optional) A character vector of region names (e.g., "Washington", "British Columbia").
year	(Optional) Integer or vector of years (e.g., 2010 or 2010:2015).
month	(Optional) Integer or vector of months (1–12). Supports wrapping (e.g., 11:2 = Nov–Feb).
valid	(Optional, default = TRUE) Filter out invalid data. Removes rows where VALID == 0.
reviewed	(Optional) If specified, filters by review status (TRUE for reviewed, FALSE for unreviewed).
rollup	(Optional, default = TRUE) Automatically roll up subspecies to species level and remove spuhs, slashes, and hybrids.

Value

A filtered dataset.

Examples

```
# Download/load example dataset
data <- pfw_example
```

```
# Filter for Dark-eyed Junco, Song Sparrow, and Spotted Towhee in Washington in 2023
data_masonsyard <- pfw_filter(
  data,
  species = c("daejun", "sonspa", "spotow"),
  region = "US-WA",
  year = 2023
)

# Filter for all data from Washington, Oregon, or California from November
# through February for 2021 through 2023
data_westcoastwinter <- pfw_filter(
  data,
  region = c("Washington", "Oregon", "California"),
  year = 2021:2023,
  month = 11:2
)

# Filter for Greater Roadrunner in California, keeping only reviewed
# records and disabling taxonomic rollup
data_GRRO_CA <- pfw_filter(
  data,
  species = "Greater Roadrunner",
  region = "California",
  reviewed = TRUE,
  rollup = FALSE
)

# Filter for Fox Sparrow with rollup
rollFOSP <- pfw_filter(pfw_example, species = "Fox Sparrow", rollup = TRUE)
# Taxonomic rollup complete. 116 ambiguous records removed.
# 1 species successfully filtered.
# Filtering complete. 8070 records remaining.

# Filter for Fox Sparrow without rollup
norollFOSP <- pfw_filter(pfw_example, species = "Fox Sparrow", rollup = FALSE)
# 1 species successfully filtered.
# Filtering complete. 7745 records remaining.

# 116 records were identified to subspecies (e.g. "Fox Sparrow (Sooty)",
# listed as 'foxsp2' in SPECIES_CODE)
# These records are merged into the parent "Fox Sparrow" total with rollup,
# but excluded in favor of records only identified exactly as
# "Fox Sparrow" (no subspecies, only SPECIES_CODE = 'foxspa') if rollup = FALSE.
```

pfw_import   *Import Project FeederWatch Data*

Description

This function reads all .csv files downloaded from the Project FeederWatch website, either from the default "data-raw/" folder created by pfw_download() or from a user-specified folder. Optionally, it can apply filters like region, species, year, etc. .csv files for import can be downloaded via pfw_download() or from the Project FeederWatch website.

Usage

```
pfw_import(folder = NULL, filter = FALSE, ...)
```

Arguments

folder	The folder where Project FeederWatch data is stored. Default is "data-raw/" in a local directory.
filter	Logical. If TRUE, applies filters using pfw_filter(). Default is FALSE.
...	Additional arguments passed to pfw_filter() for filtering (e.g., region, species, year).

Value

A combined and optionally filtered dataset containing all Project FeederWatch data.

Examples

```
## Not run:
# This example cannot be run without user-downloaded data! This data can
# be downloaded manually or with pfw_download().

# Import all downloaded data from the default folder ("data-raw")
data <- pfw_import()

# Import and filter for Washington checklists from 2023
data_filtered <- pfw_import(filter = TRUE, region = "Washington", year = 2023)

## End(Not run)
```

---

pfw_region	*Filter Project FeederWatch Data by Region*

Description

This function filters Project FeederWatch data to include only specified states, provinces, or countries.

Usage

```
pfw_region(data, regions)
```

Arguments

data	A Project FeederWatch dataset.
regions	A character vector of regions (e.g., "Washington", "United States").

Value

A filtered dataset containing only the selected regions.

Examples

```
# Download/load example dataset
data <- pfw_example

# Filter for data only from Washington using the state name
data_WA <- pfw_region(data, "Washington")

# Filter for data only from Washington using the state code
data_WA <- pfw_region(data, "US-WA")

# Filter for data from Washington, Oregon,
# and California using the state name
data_westcoastbestcoast <- pfw_region(data, c("Washington", "Oregon", "California"))
```

---

pfw_rollup   *Do Taxonomic Rollup on Project FeederWatch Data*

Description

This function removes spuhs, hybrids, and slashes and "demotes" subspecies/subspecies intergrades to their parent species.

Usage

```
pfw_rollup(data)
```

Arguments

data	A Project FeederWatch dataset.

Value

A cleaned dataset with only species-level codes and a rollup attribute.

Examples

```
# Download/load example dataset
data <- pfw_example

# Apply taxonomic rollup to an active PFW dataset
rolled_data <- pfw_rollup(data)
```

---

pfw_sitedata   *Merge Site Metadata into Project FeederWatch Data*

Description

This function joins habitat and site metadata into Project FeederWatch observation data using the site description file.If the site metadata file is not found, it will be downloaded automatically to the designated path or "data-raw" if no path is selected.

Usage

```
pfw_sitedata(data, path)
```

Arguments

data   A Project FeederWatch dataset.

path   File path to the site description .csv from https://feederwatch.org/explore/raw-dataset-requests/. If not specified, defaults to "data-raw/sitedata.csv".

Value

The original dataset with site metadata merged in.

Examples

```
# Download/loads the example dataset
data <- pfw_example

# Merge site metadata into example observation data
data_sites <- pfw_sitedata(data, "data-raw/site_data.csv")
```

---

pfw_species   *Filter Project FeederWatch Data by Species*

Description

This function filters Project FeederWatch data to include only selected species, with common names or scientific names via the species translation table.

Usage

```
pfw_species(data, species, suppress_ambiguous = FALSE)
```

Arguments

data   The Project FeederWatch dataset.

species   A character vector of species names (common, scientific, or six-letter species code).

suppress_ambiguous

                    (Optional, default = FALSE) TRUE/FALSE on including missing subspecies in the warning. This is just a silencer for the pfw_filter function.

Value

A filtered dataset containing only the selected species.

Examples

```
# Download/load example dataset
data <- pfw_example

# Filter for only Greater Roadrunner using the common name
data_GRRO <- pfw_species(data, "Greater Roadrunner")

# Filter for Lesser Goldfinch and American Goldfinch using scientific names
data_goldfinches <- pfw_species(data, c("Spinus psaltria", "Spinus tristis"))

# Filter for Dark-eyed Junco, Song Sparrow, and Spotted Towhee using species codes
data_masonsyard <- pfw_species(data, c("daejun", "sonspa", "spotow"))

# Filter with a pre-existing species list
species_list <- c("daejun", "sonspa", "spotow")
data_masonsyard <- pfw_species(data, species_list)
```

---

pfw_truncate   *Filter Project FeederWatch Data to "Standard" Seasonal Window*

Description

Project FeederWatch's Data Users Guide (https://birdscanada.github.io/BirdsCanada_PFW/Start2.html)
Suggests that data should be truncated by date to avoid biases from years where the Project Feed-
erWatch survey season was extended. This function filters data to include only observations within
the typical FeederWatch season: after November 8 and before April 3.

Usage

```
pfw_truncate(data)
```

Arguments

data   A Project FeederWatch dataset with Year, Month, and Day columns.

Value

A filtered dataset limited to Nov 8 – Apr 3 across years.

Examples

```
# Download/load example dataset
data <- pfw_example

# Truncate an active PFW dataset to November 8 - April 3
truncated_data <- pfw_truncate(data)
```

---

pfw_zerofill   *Zerofill Species not Detected in each Survey Instance for Analysis*

Description

This function adds zeros for checklists where selected species were absent, setting HOW_MANY =
0 for presence/absence-based analyses. Note that zerofilling entire, unfiltered datasets from Project
FeederWatch will take a long time!

Usage

```
pfw_zerofill(data)
```

Arguments

data   A Project FeederWatch dataset, optionally filtered for species.

Value

A dataset with zerofilled values included for each species.

Examples

```
## Not run:
# This example cannot be run because it relies on a cached version of the
# data which is created upon using pfw_import(). Storing a version of this
# for the example dataset would be too large for CRAN!

# Zerofill a PFW  dataset
data_zf <- pfw_zerofill(data)

## End(Not run)
```

---

update_taxonomy   *Update the Project FeederWatch Species Translation Table*

Description

This function downloads the latest species translation table from the Project FeederWatch website
and saves it to a local directory. If a previous version exists in the local directory, the user will be
asked for confirmation before overwriting it. This ensures taxonomy can readily be kept up to date
annually, since it will only be manually updated on the PFW website otherwise.

Usage

```
update_taxonomy(user_dir = tools::R_user_dir("PFW", "data"))
```

Arguments

user_dir   Optional. A custom directory to write the translation table to. Using the default
              local directory is highly recommended.

Value

A message confirming whether the update was successful.

Examples

```
# Prompt a species translation table taxonomy update
update_taxonomy()
```

Index