FishSET R Package User Manual

FishSET Website: https://noaa-nwfsc.github.io/FishSET/
Contact: nmfs.fishset@noaa.gov

1 Introduction

1.1 Overview

Since the 1980s, fisheries economists have employed spatial models in a variety of fisheries to better understand and explain the factors that influence the spatial behavior and fishery participation choices that fishers make when fishing. This is important for predicting how fishers may respond to, for example, marine protected areas (MPAs), climate-related species range shifts, changes in fishing costs or fish prices, fish size differences, or the implementation of various management actions such as catch share policies.

The Spatial Economics Toolbox for Fisheries (FishSET) is a set of statistical programming and data management tools developed to achieve the following goals:

- Standardize data management and organization.
- Provide easily accessible tools to enable location choice models to provide input to the management of key fisheries.
- Organize statistical code so that predictions of fisher behavior developed by the field's leading innovators can be incorporated and transparent to all users.

Modeling fisher behavior is important for designing effective fishery management policy. Failing to account for fisher behavior can lead to unanticipated consequences or failure of the management policy (Peterson 2000, Hilborn 2007, Abbott and Haynie 2012). Discrete choice random utility models(RUM), which utilize explanatory variables at the level of individual vessels to predict aggregate effort levels in alternative fishing locations, have become more widely used in fisheries (Eales and Wilen 1986, Raphael 2017, Depalle, Thebaud, and Sanchirico 2020). These models assume that fishers choose, from a discrete set of fishing locations, a location to maximize expected utility. Utility is modeled as a function of expected revenue, distance from the fisher's current location, other information about the vessel or fisher, and/or other information about the potential fishing locations (e.g. depth, temperature, spatial quota limits). Commonly used RUM techniques include conditional and multinomial logit models, nested logit models, and the mixed logit model (Train 2002). Many model variants and formulations have been developed by economists in fisheries and other fields, with innovations motivated by the policy question being analyzed, the nature and extent of available data, computing power increases, and the statistical background of the analysts.

1.2 Why use FishSET?

FishSET is intended for all users, regardless of experience with R or any other programming language.

FishSET includes an easy-to-use graphical user interface that guides users through the steps from loading data to evaluation models without writing any code.

For users with experience in R who need more flexibility than the graphical user interface provides, functions can be run directly in the R console.

Regardless of whether FishSET functions are run in the graphical user interface or the R console, users must have R and RStudio installed.

Instructions to install R and RStudio and the FishSET package (a collection of tools or functions) are provided in this manual along with details on using the package.

The intent of FishSET is to provide tools which improve both the quality of management analyses and the ease of conducting analyses. The FishSET R Package provides a wide range of tools, including functions for:

Data Management

Good data management can save time over the duration of a project, prevent errors, increase quality of analyses, and allow for validation and replication of findings. In FishSET, all data files are automatically saved to a single location, a SQLite database called the 'FishSET database'. Changes to the data file are saved in the database as a new table which is then updated; the original loaded data is never modified.

Data Analysis

FishSET provides functions, grouped into steps, which guide users through important steps in checking and addressing data quality, understanding the variance, distribution, and availability of data through data exploration, and the steps to generate derived variables (e.g., zonal averages, variable interactions, or calculated variables such as CPUE) that may be needed in analyses.

Visualizing data

Creating figures from fisheries data is a critical part of examining and summarizing data. FishSET provides numerous tools to view the spatial distribution of vessels, hauls, or other variables of interest, including hotspot analysis and haul paths. Visualization at various levels of aggregation, including by time, location, fishery, or vessel, or other characteristics are also possible.

Statistical modeling of fisher behavior

FishSET includes functions to estimate location choice models. FishSET allows users with knowledge of the models to use location choice models developed by different researchers. FishSET also enables comparisons of numerous model specifications, model assumptions, and model types.

Policy comparison

After behavioral models are developed for a fishery, FishSET allows comparisons of actual or simulated fisheries policies, such as the creation of spatial closures.

Reproducibility

Function calls in FishSET are logged and saved in a dated file within the FishSET R package directory. The log function calls save the function name, the parameter values, including input data, and output. Function messages, such as the presence of missing values, are also stored. Storing messages provides a record of the data that informs choices, such as removing rows from the data that contain missing values. This means that all steps in the analysis are automatically saved and the analysis can be quickly and easily reproduced at any time and/or rerun with updated data.

1.3 FishSET R Package

The FishSET R package (FishSET) is a collection of tools or functions written in R for managing and analyzing fisheries data that have been grouped in a sharable format (package). Functions focus on six key areas:

- Data management,
- Data analysis and mapping,
- Model preparation,
- Model design and evaluation,
- Policy design and evaluation, and
- Reporting.

Although users must have R installed on their computers and must work in R, we do not assume or require knowledge of R or programming abilities, and FishSET includes a graphical user interface (FishSET GUI) that guides users through the steps to develop a location choice model. This manual and the FishSET GUI include a quickstart guide that explains how to use each tab of the GUI. This method is a user-friendly approach for users not familiar with R or with location choice models.

Alternatively, users can execute FishSET functions within the R console. There are benefits to working in the R console, namely greater control of functions, the ability to run functions and analyses beyond what is included in FishSET, and easier troubleshooting. However, working in the R console does require some knowledge of R programming.

FishSET automates many important but often overlooked steps. These include saving the original data files along with modified and finalized data tables, saving all table and plot outputs, messages generated by analyses, notes, and record of function calls. The automated saving of files, output, and function calls is designed to enhance transparency and reproducibility (see Chapter 3 for more details).

2 Quickstart Guide

This chapter describes how to install the software and R packages (bundled sets of code) necessary to load the FishSET R package. A minimal set of functions are shown to demonstrate how to use FishSET in the R console and in the FishSET GUI.

2.1 Install software

R and RStudio are required software for FishSET. In addition, RTools is required for Windows users in order to install the FishSET package and various other packages will be installed in the process.

2.1.1 Install R (required)

Install R from https://cran.rstudio.com/. R should be installed prior to installing RStudio. You will not need to open R independently if using RStudio.

2.1.2 Install RStudio (required)

Install RStudio Desktop from https://www.rstudio.com/products/rstudio/download/#download. RStudio provides an easy to use interface for programming in R.

2.1.3 Install RTools (required for Windows users)

Install RTools for Windows from https://cran.r-project.org/bin/windows/Rtools/rtools40.html.

It is necessary to put Rtools on the file PATH. To do this, open RStudio and paste the following into the R console window.

```
write('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"',
file = "~/.Renviron", append = TRUE)
```

Restart R before continuing. In RStudio this can be done by clicking the Session tab and selecting Restart R from the dropdown menu.

2.2 Installing FishSET R package

There are currently two methods for installing the FishSET R package: (1) installing the package from the FishSET GitHub repository and (2) installing from a .tar file. Each method is described in detail below.

2.2.1 GitHub install

Installing FishSET through the GitHub repository is the preferred method because users will have access to the latest version of FishSET and can reinstall updates on their own.

Go to the FishSET GitHub repository and follow the instructions for GitHub installation.

2.2.2 Local install

If users are unable to install FishSET from the GitHub repository, the package can be installed using a .tar file provided by FishSET personnel (contact nmfs.fishset@noaa.gov). The disadvantage of this method is that updates to the package are not readily available to the user unless an updated .tar file is sent.

Go to the FishSET GitHub repository and follow the instructions for local installation.

2.2.3 Troubleshooting install

Error message	**Action**
`Do you want to install from sources the package which needs compilation?`	Type “Y” or “Yes” and hit enter.
`package ‘fishset’ is not available (for R version x.y.z)`	Package misspelled or case incorrect. The package must match ‘FishSET’ when installing.
`Error: dependency ‘foo’ is not available for package “FishSET” * removing ‘C:/R/win-library/4.1/FishSET’...`	Missing a package dependency or dependency needs to be updated. Install package ‘foo’ then install FishSET. Note ‘foo’ is a placeholder name for troubleshooting purposes.
`Error: package or namespace load failed for ‘FishSET’: .onLoad failed in loadNamespace() for ‘rJava’, details:...`	Java software on your computer and R (32/64 bit) do not match. Run `system("java -version")` to find Java version and change in RStudio by clicking on Tools > Global Options and change R version. Then restart

	RStudio.
`Error in utils::download.file(url, path, method = method, quiet = quiet,...`	Run the following line of code: options(download.file.method = "wininet"). Then run remotes::install_github.
`Error in dyn.load(file, DLLpath = DLLpath,...) unable to load shared object...`	Reinstall the package indicated in the error message and restart the R session. If issues persist, try uninstalling and reinstalling R and R studio.
`Error: failed to lock directory...`	Lodate and delete the ".../00LOCK-[packagename]" and the "[packagename]" folders in the library folder, which should be displayed with the error message, then attempt to reinstall the problem package.

2.3 Navigating FishSET

Prior to using the FishSET package, create a folder on your computer labeled 'FishSETFolder'. All project data, analyses, etc. will be saved in this directory. Users can run FishSET functions in the R console or through the FishSET GUI (graphical user interface). For users new to R or to location choice modeling, we recommend starting with the FishSET GUI. The GUI guides users through the steps from loading data to running policy scenarios without requiring any knowledge of coding in R. However, the R console allows greater flexibility.

The first steps, using either the FishSET GUI or the R console, are defining a project name and arranging your data in the required format (see Chapter 4 Data). The project name is required for all FishSET functions and used to distinguish data, plots, tables, and other outputs between distinct projects or analyses. Projects can be named by fishery, year, location, goal, or any other meaningful character string.

In the next sections of this chapter, we outline a limited set of functions that will take users from loading the data to running models. The FishSET website contains a complete list of functions with documentation for each function.

2.3.1 Getting help with functions

Function documentation provides details on functions including required and optional input arguments, how to specify/format inputs, and a description of the output. Some function documentation also includes examples of using the function in the console. There are three ways to access function documentation. First, go to the FishSET website and navigate to the reference tab. Second, query a specific function by typing `?FUNCTION_NAME` in the R console, replace `FUNCTION_NAME` with the name of the function you want to explore and documentation will appear in the *Help* viewer pane. Third, in RStudio click on FishSET in the *Packages tab* (arrows in Figure 2.1a). The documentation will then appear in the *Help* tab (arrow in Figure 2.1b).

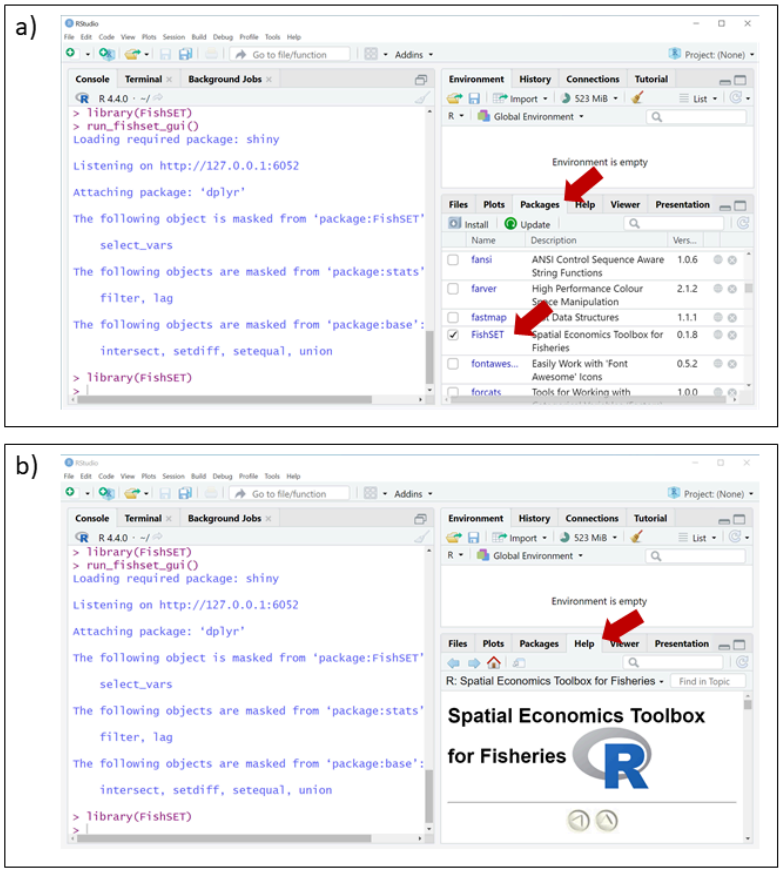


Figure 2.1 Documentation for packages can be obtained by clicking the package name in the Packages tab (a) and then viewed in the Help tab (b)

2.3.2 Navigating FishSET functions in the R console

One option for starting with FishSET in the R console is to follow the vignette developed for this purpose. This vignette uses a sample fictitious dataset ("scallop", which is available to all users after installing the FishSET package) to demonstrate core FishSET functions.

2.3.3 Navigating FishSET using the FishSET GUI

In the R console, open the FishSET GUI by running:

```
#Load FishSET
library(FishSET)

#Start FishSET GUI
run_fishset_gui()
```

To get started in the FishSET GUI, follow the Quick Start Guide. To use the sample scallop dataset in the GUI, save them locally (note that .rds files below will save to the current working directory in R):

```
library(FishSET)
saveRDS(scallop, "scallop.rds")
```

```
saveRDS(scallop_ports, "scallop_ports.rds")
saveRDS(tenMNSQR, "tenMNSQR.rds")
```

and upload them to the GUI following the GUI Quick Start Guide.

3 Saving Inputs and Outputs

3.1 FishSETFolder directory

All data and outputs are stored and organized in a directory called the 'FishSETFolder'. This directory is important for maintaining a transparent and reproducible workflow. The directory only has to be created once, and the location of the folder on your computer can be changed.

The 'FishSETFolder' directory should not be renamed because FishSET functions are written to search for the path containing the directory. If the directory and subfolder names are changed, FishSET functions will not be able to locate the data needed for analyses or outputs needed to generate reports.

FishSET creates subdirectories within the FishSETFolder for storing data and output, organized by projects. Each time a new project is identified, either in the FishSET GUI or in a function call, a new project subdirectory will be created. For each project, FishSET will automatically create a local SQLite database (fishset_db.sqlite) and create five subdirectories (src, output, MapViewer, doc, and data), which are described in more detail below. Note that data are only stored locally on the user's computer and will not be shared with FishSET developers or other FishSET users.

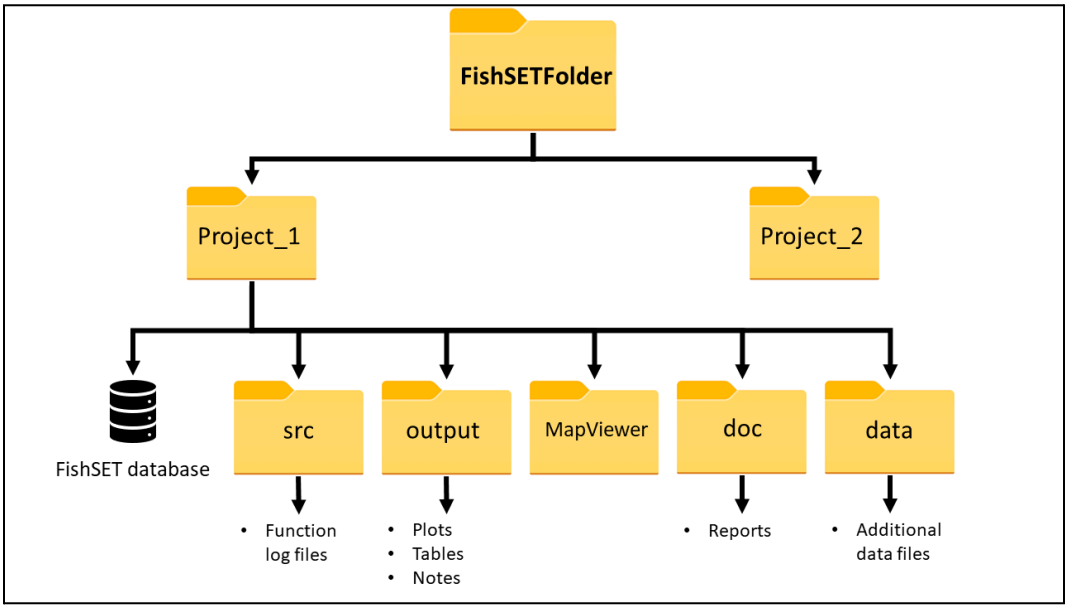


Figure 3.1 FishSET package folder structure.

3.1.1 Projects

Inputs and outputs are organized into *Projects* (Figure 3.1). A *project* is defined by the user but should indicate a distinct set of analyses. Projects can be based on species, areas, years, a combination of species, area, and years, or any other unique identifier that defines the scope of the work you are doing.

The project name can contain letters, numbers, or characters, but should be meaningful to you and distinct from other projects. The creation of *project* subdirectories is automated by FishSET.

3.1.2 FishSET database

Within each project folder there is a *FishSET database* where all data are stored. The *FishSET database* is saved as a local SQLite database (fishset_db.sqlite), which is a light-weight, self-contained, single-user SQL database engine. Both the **RSQlite** and **DBI** R packages are required to communicate with the *FishSET database*. Because the SQL code is embedded into functions it is important to not move or rename the *FishSET database*.

The name of the primary data, which holds information on trips or hauls, contains the string 'MainDataTable'. Modified versions of the data table contain the original name plus the date it was saved. All tables in the *FishSET database* are saved with the project name. There are rules for the names of other data types (outlined in Chapter 4) and model output data tables.

3.1.3 src folder

Every function call, whether run in the R console or through the FishSET GUI, is saved or logged in the *src* folder. Each logged function call contains the function name, all defined parameters, and any informational messages. Functions are logged in the order they are called. The logged function call also serves as a record of steps taken, which can be used when writing reports. The log_rerun() function can be used to rerun the set of logged function calls with updated data.

3.1.4 output folder

All plots and tables created within function calls are automatically saved to the *Output* subdirectory. Output files are saved with the project name, the function name, and the date it was generated. Plot and table files will be overwritten if a function is rerun on the same day. To bypass this default behavior, you must save output to an alternative, user-defined location or rename the saved output. Plots are saved as .png files whereas tables are saved as .csv files.

Informational messages generated by functions are saved to a text file in the *Output* subdirectory and in the msg section of function call logs. Informational messages provide a record of results from checking data quality and of potential data issues. Saving these messages is useful for keeping track of result-driven decisions. For example, a choice to remove the vessel_length variable after finding that 90 of 100 data points were empty. In the FishSET GUI, notes and comments written by users are saved with the function-generated informational messages.

3.1.5 MapViewer folder

The MapViewer folder contains code for running the MapViewer function in the FishSET GUI. Do not make any changes or add any other files/folder to the MapViewer folder.

3.1.6 doc folder

The doc folder contains an RMarkdown template to write a report, and the output is stored in the *doc* subdirectory. DEVELOPMENT NOTE: this feature is still under development (6/20/2024).

3.1.7 data folder

Text files associated with the data files (e.g., metadata file) and data files that cannot be saved to the FishSET database (e.g., .geojson files) are saved in the *data* subdirectory.

4 Required and Optional Data

4.1 Data Types

FishSET is designed to parse and save datasets used in analyses. Details on data storage are provided in Chapter 3. Required data include the primary and spatial data files. The port data file is optional, as port data can be included in the primary dataset. Other optional data include gridded data and auxiliary data (Table 4.1). The port table, gridded data, and auxiliary data must be able to be linked to the primary data table by a common variable. For example, the port table must contain a variable that identifies individual ports that matches a port identifier variable in the main data table (e.g., a port name variable in each dataset).

Table 4.1 FishSET data file types and their description.

Data type	Requirements	Description
Primary data	Required	Main data for models and simulating policies.
Port data	Optional	Contains port name, latitude, and longitude. Can be included in primary data.
Spatial data	Required	Defines boundaries of fishing zones. Can also define spatial closures.
Gridded data	Optional	Supplemental data that varies by zones (can also vary by time).
Auxiliary data	Optional	Additional data that links to the primary data by a variable in common between the datasets.

4.1.1 Primary data file

The primary data is a flat file containing the core information used in models. This data file must contain a trip ID, date/time, catch and/or revenue, fishing location, and port location or identifier if using a port table (Table 4.2). If not using a separate port table, port information must be included in the primary data. Additional information such as price, species caught, and vessel characteristics may be included in the primary or the auxiliary data.

Each row of the primary data file should be a unique observation and each column a unique variable. Single or double apostrophes, commas other than as delimiters, and periods other than as decimal points, should not be included in the file. Use underscores rather than spaces in column names and use NA or leave cells empty to represent no data.

Required variables

Table 4.2 Description of required variables in the primary dataset.

Variable	Description
Port	At least one port variable must be included, such as disembarked or embarked port. Ports can be represented by numerical or character IDs. Both port ID and port name variables may be included, but at least one will need to be in common with the port data file if uploaded.

Full date	Several date variables may be included, such as haul date, trip start and end dates, and fishing year. At least one date variable should be in complete date format (e.g. year, month, and day); hour, minutes, and seconds are optional.
Catch	At least one catch variable should be included. Catch data must be in a measure of weight (e.g., metric tons, kilograms). Catch count may also be included but must be in addition to weight.
Fishing locations	Fishing locations are preferably defined by latitude and longitude (separate columns and in decimal degree format.) If there are no lat/lon fishing locations then there must be a variable which associates fishing location to known fishery management or regulatory zones. In this case, the centroid of fishery management or regulatory zones will be used as fishing locations. See spatial and mapping functions in Chapter 6 Exploratory data analysis for more information on how to assess and address fishing location issues.

In addition to the required variables above, additional variables may be required to link or merge the primary data file to the port, auxiliary or gridded data files. For example, if the auxiliary data contains information on vessel characteristics then both the auxiliary and primary data files must contain the vessel identifier variable.

Optional variables

Optional variables can be almost anything important to your data set. Examples include price data , vessel information, or gear type. In addition, you can include multiple port, date, catch, and fishing location variables. For example, the primary data file in Figure 4.1 contains disembarked and embarked ports, trip and haul starting and ending dates, catch variables for two species, and starting and ending fishing locations.

Representing ID data

Identification data (such as individual vessels, ports, and gear type) can be included in the primary data file in the following ways:

A. Use a single code variable containing the ID's for the variable. In this case, no description will be associated with the ID and only the ID will be used for plotting and labeling figures. VESSEL_ID is treated in this way in the primary data file example (Figure 4.1).

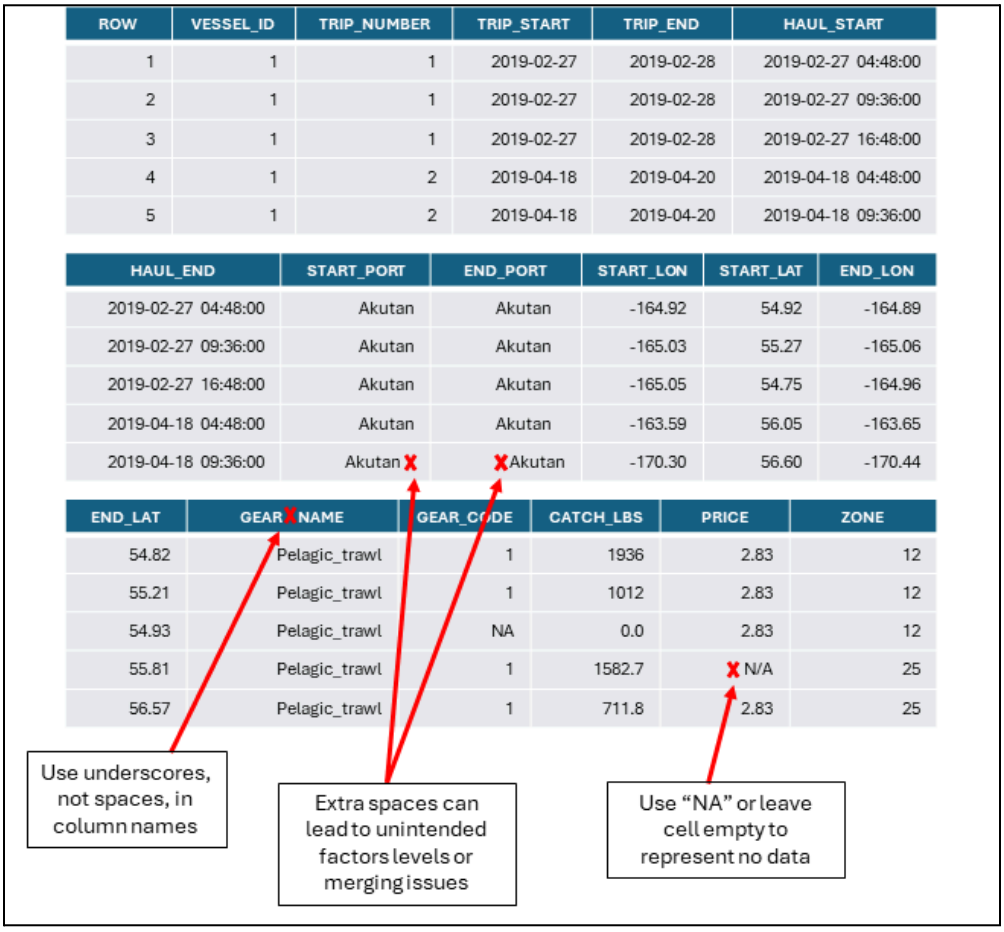
B. Use a single string variable containing the ID descriptions. In this case, FishSET will internally assign each unique description a numeric ID, but the string description will be used for plotting and labeling figures. In Figure 4.1, GEAR_NAME is an example of a string variable.

C. Include two variables, one code variable containing the ID's and one string variable containing the ID descriptions. Gear type is treated this way in the primary file example (GEAR_NAME and GEAR_CODE) (Figure 4.1).

Row level data

The primary data file consists of variables as columns, and observations as rows. Each row should be a unique choice occurrence at the set/haul or trip level. A haul-level data file may include several rows of data for one trip, with each row being a unique haul for that trip (as in Figure 4.1). A trip-level data file would only have one row of data for each trip. For both haul-level and trip-level data files, each row

should be a unique choice occurrence. A trip identifier variable can also be created using FishSET functions. For instance, in Figure 4.1, TRIP_NUMBER is created from VESSEL_ID and TRIP_START.



ROW	VESSEL_ID	TRIP_NUMBER	TRIP_START	TRIP_END	HAUL_START
1	1	1	2019-02-27	2019-02-28	2019-02-27 04:48:00
2	1	1	2019-02-27	2019-02-28	2019-02-27 09:36:00
3	1	1	2019-02-27	2019-02-28	2019-02-27 16:48:00
4	1	2	2019-04-18	2019-04-20	2019-04-18 04:48:00
5	1	2	2019-04-18	2019-04-20	2019-04-18 09:36:00

HAUL_END	START_PORT	END_PORT	START_LON	START_LAT	END_LON
2019-02-27 04:48:00	Akutan	Akutan	-164.92	54.92	-164.89
2019-02-27 09:36:00	Akutan	Akutan	-165.03	55.27	-165.06
2019-02-27 16:48:00	Akutan	Akutan	-165.05	54.75	-164.96
2019-04-18 04:48:00	Akutan	Akutan	-163.59	56.05	-163.65
2019-04-18 09:36:00	Akutan	Akutan	-170.30	56.60	-170.44

END_LAT	GEAR_NAME	GEAR_CODE	CATCH_LBS	PRICE	ZONE
54.82	Pelagic_trawl	1	1936	2.83	12
55.21	Pelagic_trawl	1	1012	2.83	12
54.93	Pelagic_trawl	NA	0.0	2.83	12
55.81	Pelagic_trawl	1	1582.7	N/A	25
56.57	Pelagic_trawl	1	711.8	2.83	25

Figure 4.1 The primary data file at haul-level with examples of both required and optional variables. 'X' indicates examples of unacceptable input formats.

4.1.2 Port data file (optional)

The port data file contains the location of ports in the primary data file. Port location data is required for several functions in FishSET that calculate or use distance between points. Port data should contain, at a minimum, the port name, latitude, and longitude (Figure 4.2). Each row of the port data should be a unique port. Port names may be numeric identifiers, or string, as long as port identifiers match between the port and primary data files. The port identifier variable, which links the port data to the primary data file, must be identified when port data is loaded into the FishSET database. Latitude and longitude must be in decimal degrees with cardinal direction indicated by sign. Common errors are shown in Figure 4.2.

PORT_NAME	PORT_LAT	PORT_LON
Akutan	54.135	165.773 W
Akutan	54.135	- 165.773

Dutch Harbor	53° 53'23''	- 166.542

Figure 4.2 Example of Port Data. 'X' indicates examples of input format errors, including preceding spaces, using letters rather than signs to indicate cardinal direction, and specifying latitude and longitude in a format other than decimal degrees.

.4.1.3 Spatial data file

The spatial data file contains information on fishery management or regulatory zones. It is a spatial data file containing the latitude and longitude points defining zone polygons. The file can be a shapefile, JSON, GeoJSON, data frame, or list (the last two can be saved and uploaded as R .rds files).

The spatial data file is essential and used to assign observed vessel locations to zones, calculate zonal or fishing centroids, and calculate distance matrices. The spatial file must contain a zone identifier and latitude and longitude points defining zone boundaries. Each zone that you want to consider separately should have a unique ID. Multiple zones with the same ID will be combined into one zone and treated as such in FishSET, even if that means multiple parts of a zone are separate in space (Figure 4.3). FishSET imports spatial data files as is, even if the zone outlines cover land. The land will NOT be subtracted out by FishSET; this is up to the user.

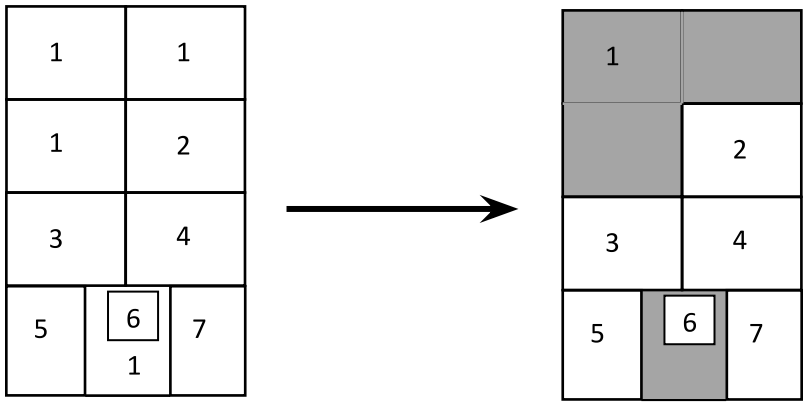


Figure 4.3 Multiple zones with the same ID are treated as a single zone.

4.1.4 Gridded data file (optional)

A gridded data file contains a variable associated with zones in the spatial file. It can also vary by a second dimension (e.g., date/time). Gridded data can be used in models and in calculating alternative fishing choice matrices. Examples of gridded data include sea surface temperature, wave height, ice cover, and wind speed.

Columns of the gridded data file are defined by grid locations (zones) that match zones in the primary data file. In Figure 4.4, the column names match the values in 'Zone' in Figure 4.1. The optional second dimension is represented by rows, and the row names must exactly match values of the corresponding variable in the primary data file (TRIP_START in Figure 4.4). The gridded data file should not contain single or double apostrophes, commas other than as CSV separators, or periods other than decimals. Missing data should be indicated with NA or an empty cell.

TRIP_START	745830	75830	715830	705830	69830
04/01/2019	2.95	2.32	2.37	2.81	2.13
04/02/2019	2.90	2.49	2.86	2.25	2.60
04/03/2019	3.19	3.41	3.25	4.00	3.08
04/04/2019	1.73	1.08	1.03	NA	1.10
04/05/2019	2.17	2.50	3.00	2.15	2.13

Figure 4.4 Wind speed data by NMFS Statistical Area and trip start date.

4.1.5 Auxiliary data file (optional)

Auxiliary data files can contain anything you want to merge with the primary data file. Thus, the auxiliary data must contain a variable in common with the primary data. Auxiliary data are especially useful for storing information that can be used across multiple primary data files, such as vessel characteristics. It is also an efficient way to include data that are not haul- or trip-level specific. Examples of auxiliary data include vessel characteristics, prices by date, and fishing season data. Auxiliary data can also be used to generate new variables, such as fishing season identifiers.

An auxiliary data file is set up similarly to a primary data file but is not haul- or trip-level specific. In Figure 4.5, the merge variable is 'VESSEL_ID' (primary data must also contain this variable). As with gridded and port data files, it is important to ensure spelling, capitalization, preceding and following spaces, and characters exactly match between the merge variable in the primary and auxiliary data files.

VESSEL_ID	VESSEL_LENGTH	VESSEL_HP
1	117	1148
2	138	1180
3	NA	NA
4	66	539
4	66	539

Figure 4.5 Auxiliary data file containing vessel characteristics.

4.2 Upload data into FishSET

FishSET includes functions to read data into R, parse the data, and then save the data to the FishSET SQLite database. This process ensures that data meet the minimal requirements outlined in Section 4.1 Data Types. For instance, the primary data table contains information on catch, ports, and location. It also ensures that along with the data file that will be used in analyses a second copy is made that will not be altered.

4.2.1 FishST GUI

In the FishSET GUI, all data are loaded using the Upload Data tab. You can browse to the file location of each data type (primary, port, spatial, etc.) before clicking on the 'Load Data' button. A project name must be specified prior to uploading data. This process can be repeated to load additional auxiliary, gridded, or spatial data.

4.2.2 R console

Data should be imported into FishSET using the load functions (`load_maindata`, `load_port`, `load_spatial`, `load_aux`, `load_grid`). Each of the load functions reads a specific type of data into R and saves the data in a local SQLite database. If data are already in the FishSET database, use `load_data` to load the data into the working environment.

4.2.3 Data preparation

When working with R, there are a few simple rules to make data importing easier and reduce potential errors, especially for merging data. Here are some common mistakes for data in spreadsheet format.

Spreadsheet format

Data saved as .csv, .txt, .xls, .xlsx, or google sheets are generally in a spreadsheet format. These data will be converted into data frames in R.

- The first row should be the header and each row should be a unique observation.
    - Delete any comments. Including comments can result in extra columns or NAs being added. Metadata should also be deleted and saved in a separate file or in an R script file. If metadata is not deleted, you will need to skip the number of rows containing metadata when data is imported.
    - Missing values should be indicated with NA or left empty. Do not delete missing values or replace them with 0. A value, even if empty, must be provided for each column for every row of the data. Mishandling of missing values can result in a 'number of elements per line' error when importing.
- Avoid blank spaces in names. When importing, each word separated by space will be interpreted as a separate variable. This can result in a 'number of elements per line' error when importing and difficulties with merging datasets.
    - Rather than using spaces in names, use an underscore or camel case to concatenate words.
- Names should be informative and as short as possible.
- Avoid using symbols in names such as $, %, ^, *, (, -, #, ?, ,, <, /, |, \, [,], {, and }. These, and other symbols, are operators which allow use of arithmetic expressions.

4.2.4 Accepted file extensions

FishSET recognizes how to read in a wide range of file formats. Data file types not listed in Table 4.3 can be used but will require the user to specify the external function (from a package other than FishSET) to read in data.

When a dataset is read into R using either the load functions or `read_dat`, the function detects the file format and then calls the appropriate function from another R package to convert the file into a format recognized by R.

If data are not in the basic format, such as a tabular structure with no rows containing metadata for csv, then an error message may occur and additional arguments will be required to inform the function how to read in the data. For example, for a csv file that does not contain column headings, we would add `headers = F`. If, in addition, the first three lines of the data file contained metadata, we can specify to skip those lines with `skip = 3`. The function and inputs to accommodate this would be:

```
read_dat('C:/data/dat.csv', header = FALSE, skip = 3).
```

There are numerous arguments that can be added to address common errors that occur when reading in data. These arguments can also be used in the FishSET GUI.

File extension	Function	Package	Link
.csv	read_csv	readr	https://readr.tidyverse.org/reference/read_delim.html
.geojson .gpkg	st_read	sf	https://r-spatial.github.io/sf/reference/st_read.html
googlesheets	read_sheet	googlesheets4	https://www.rdocumentation.org/packages/googlesheets4/versions/0.1.1
.html	read_html	xml2	https://www.rdocumentation.org/packages/textreadr/versions/1.2.0/topics/read_html
.json	fromJSON	jsonlite	https://www.rdocumentation.org/packages/jsonlite/versions/1.7.1/topics/toJSON%2C%20fromJSON
.mat	readMat	R.matlab	https://www.rdocumentation.org/packages/R.matlab/versions/3.6.2/topics/readMat
.ods	read_ods	readODS	https://www.rdocumentation.org/packages/readODS/versions/1.7.0/topics/read_ods
.rds	read_rds	readr	https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/readRDS
.sas7bdat	read_sas	haven	https://haven.tidyverse.org/reference/read_sas.html
.sav, .sps, .por	read_spss	haven	https://www.rdocumentation.org/packages/haven/versions/1.0.0/topics/read_spss
shape	st_read	sf	https://r-spatial.github.io/sf/reference/st_read.html
.dta	read_dta	haven	https://haven.tidyverse.org/reference/read_dta.html
SQL	dbConnect,db GetQuery	DBI	https://dbi.r-dbi.org/
.txt	read_tsv	readr	https://readr.tidyverse.org/reference/read_delim.html
.xls,.xlsx	read_excel	readxl	https://readxl.tidyverse.org/
xml	read_xml	xml2	https://www.rdocumentation.org/packages/xml2/versions/1.3.3/topics/read_xml

Table 4.3 Functions and packages for each file extension called by the FishSET read_dat() function.

4.3 Managing the FishSET Database

FishSET uses a SQLite database to store data, and several different tables will be created in the database during the course of an analysis. Table 4.4 shows each table type, its naming convention, and a description. Each table name begins with its project name, e.g. "*project*MainDataTable".

Table Name	Description
MainDataTable	Primary table used in analysis
MainDataTable_final	Final version of primary table used in models
PortTable	Table containing port name, longitude, and latitude
SpatTable	Spatial table used to define closure areas and zones
GridTable	Data that varies by regulatory zone
AuxTable	Secondary data that can be joined to primary table
FilterTable	Filter conditions used on primary table
FleetTable	Fleet definition table
ZoneCentroid	Zonal centroid table
FishCentroid	Fishing centroid table
AltMatrix	List of alternative choices
ExpectedCatch	List of expected catch matrices
ModelInputData	List of defined models and data
ModelOutput	Model output such as estimates, SE, etc.

ModelFit	Table of model comparison metrics
LDGlobalCheck	Model error output

Table 4.4 List of FishSET database tables and their description.

FishSET provides several table management functions as well.

Purpose	Function
List all tables for a project	tables_database()
List all fields (columns) in a table	table_fields()
Load a table into R console	table_view()
Delete a table from a project	table_remove()
Save a main, port, spatial, gridded, or aux table	table_save()
Check if a table name exists	table_exists()
List table names by type	list_tables()
Load the alternative choice list to R console	alt_choice_list()
Load the expected catch list to R console	expected_catch_list()
Load the model design list to R console	model_design_list()
Load model output list to R console	model_out_view()
Load model parameters to R console	model_params()
Load model fit metrics to R console	model_fit()

Table 4.5 List of functions for navigating and managing the FIshSET database.

4.4 Cleaning and checking data

FishSET read and import functions will import data but will not resolve issues that may cause import failure, that make it hard to extract meaningful results, or that lead to spurious results. The next sections in this chapter outline modifications that must be made to the data before import. We then outline how to tell if your data is messy and, therefore, likely to make working with your data more challenging. Finally, we outline a set of data quality checks that should be applied before any analyses are done. For tips on handling large datasets, missing values, etc. see Chapter 11 Tips.

4.4.1 Data format requirements

This section outlines issues that can lead to errors when working with the data or when the data is saved to the FishSET database. R requires that data frames be a single, flat data table with variables in columns and observations in rows.

It is advisable to perform some basic data cleaning before loading the data file into R. SQLite is case-insensitive. Row names are not required but should be specified if they exist in the data file. Numeric variables can contain only numeric data and any symbols, letters, or spaces must be removed from these columns. The unit of measurement should be stored in the column name, a separate variable, or in a separate metadata file. The default decimal separator is a point. If a different decimal separator is used but not specified at import, the numeric values will be treated as characters. If any characters are found in a numeric vector, the vector will be classed as a character vector at import and converted to a factor when used in statistical functions. Converting the vector to numeric (`as.numeric()`) before applying a statistical function may not be sufficient as all cells with characters will return NA.

4.4.2 Identifying messy data

Messy data can make it hard to extract the necessary information and require extra steps to make the data usable. Also, messy data can result in overly large datasets that slow down basic data visualization functions (plotting, tables) for data checking and cleaning. For more information on tidy data in R review Chapter 12 Tidy Data in R for Data Science by Wickham and Grolemund.

4.4.3 Data quality checks

FishSET provides a number of statistical and graphic functions for evaluating data quality for user convenience. These functions focus on issues that are most likely to affect model convergence and model performance. They cannot uncover all data quality issues that may exist in your data. The user must ensure that the data are of sufficient quality for any subsequent analysis.

Data quality checks

A number of checks are completed when data are imported and saved to the FishSET database. Data are checked for unique, case-insensitive variable names, that each row is a unique observation at the haul or trip level, that there are no empty variables, and that latitude and longitude variables are included. Duplicate rows and empty variables can be fixed after data has been loaded. However, duplicate variable names and missing latitude and longitude must be addressed by the user before the data can be saved to the FishSET database.

Variable class

We recommend checking that variables are classified correctly after uploading data. See Chapter 11 Data Import in R for Data Science by Wickham and Grolemund for more information about data classes in R.

Data summary

The summary table will display the statistical summary for numeric variables. Check these to ensure that minimum and maximum values are constrained to possible values (e.g., no negative vessel hauls). This information can also provide an indication that an error is present in the data, such as a decimal point in the wrong place, error in data entry, or inconsistent units within a variable. Other potential sources of data entry errors should also be checked, including duplicate observations and variables that contain no data. These sources of error should be removed.

Missing values

Missing values (NA) and non-numbers (NaN) can cause problems with computations, and certain FishSET functions will not run when NAs or NANs are present in the primary data. Missing values for numeric variables should not be replaced with 0. FishSET provides the option to remove or replace all rows containing missing values or non-numbers from the dataset. See Chapter 11 Tips for more information on handling missing values.

Outliers

FishSET provides a number of descriptive statistics to assess whether any data points are outliers, but does not provide formal techniques to detect outliers (see the **outliers** or **OutlierDetection** packages). FishSET provides functions to visualize the distribution of a variable and the effect of removing points with values that exceed frequently used definitions of extreme values (Table 4.5).

Latitude and longitude format

Latitude and longitude variables must be in decimal degrees. Western longitudes and southern latitudes must be preceded by a negative sign. Letters, spaces, and other characters are not allowed. FishSET provides a function to check and correct any issues.

Functions

FishSET provides several functions to check for and correct common data quality issues. FishSET also includes the data_verification() function, which calls the functions that are applied to the data table rather than variables in the data table. To view help documentation for each function in Table 4.6, go to the FishSET GitHub webpage.

Purpose	Function
Runs several checks, including empty variables	data_verification()
Check and change variable data class	changeclass()
Remove empty variables	empty_vars_filter()
Remove or replace NAs	na_filter()
Remove or replace NaNs	nan_filter()
Remove duplicate rows	unique_filter()
View summary statistics	summary_stats()
Box-and-whisker plot of all numeric variables	outlier_boxplot()
Assess potential outliers in table format	outlier_table()
Assess potential outliers in plot format	outlier_plot()
Remove outliers	outlier_remove()
Check and correct lat/lon format	degree()
Check and correct spatial issues	spatial_qaqc()

Table 4.6 List of functions for checking for and correcting common data quality issues. View documentation for each function on the FishSET GitHub webpage.

FishSET GUI

In the FishSET GUI, all data quality functions mentioned are located in the *Data Quality Evaluation* tab.

4.5 Data Integration

Secondary data (auxiliary, port, gridded) can be merged into the primary dataset. The default method to merge is to use a "left join"; all columns and rows from the primary dataset are kept whereas only matching columns and rows from the secondary table are joined. Although all secondary data types can be merged into the primary dataset, generally only auxiliary data (such as vessel characteristics) will need to be merged.

Secondary data can also be split from the primary dataset and saved in an auxiliary data table in the FishSET database.

Users should use the data integration functions to merge primary data that comes from several sources, such as:

- Self-reported (log book, fish ticket, trip report, etc.)
- Observer data or electronic monitoring
- VMS, AIS

- Model estimated fishing
- Electronic recording of fishing (e.g. winch monitor).

4.5.1 Functions

FishSET provides functions to merge data and to split data (Table 4.7). To view documentation for these functions, visit the FishSET GitHub webpage.

Purpose	Function
Merge secondary data into the primary data table	merge_dat()
Split and save secondary data from the primary data table	split_dat()

Table 4.7 List of functions for merging and splitting data.

FishSET GUI

Auxiliary data can be merged into the primary dataset on the `Upload Data` tab. Load the primary and auxiliary data from the desired source. Check the `Merge auxiliary table` box. Select the variables (table keys) from the primary/main and auxiliary data tables to merge by. Finally, press the blue `Merge` button.

Data can be saved to the FishSET database or to an external file directory at any time. FishSET includes two functions to save data (Table 4.8). The save_dat() function saves the data to the FishSET database whereas the write_dat allows users to specify where to save the data and the file extension. To view documentation for these functions, visit the FishSET GitHub webpage.

Purpose	Function
Save data to the FishSET database	save_dat()
Save data to specified location and file format.	write_dat()

Table 4.8 List of functions for saving data.

FishSET GUI

Data are saved to the FishSET database in the FishSET GUI by clicking the *'Save data to FishSET database'* button. This button is present in the top left-hand corner of each tab of the FishSET GUI that has functions that modify the primary data table.

4.6 Working with confidential data

The Magnuson-Stevens Act requires a signed non-disclosure agreement to release confidential data. Otherwise, data must be aggregated or summarized and pass further checks that confidential data could not be derived from aggregated data.

FishSET does not provide guidance on which checks should be used as there is not a standardized set of rules. However, FishSET does provide two of the most commonly used rules, the "rule of n" and "identification of majority allocation" rule. The *rule of n* requires a minimum reporting size (`n`) to display aggregated data. The *identification of majority allocation* requires that no single data point can be identified as making up the majority of the aggregated or summarized value. Aggregated values that do not meet these requirements are suppressed. Values that include the suppressed values may also be suppressed (complimentary suppression) so that suppressed values cannot be easily derived.

4.6.1 Functions

FishSET includes a function to define confidentiality rules that will be applied to all table and figure output (Table 4.9). Two additional functions are also provided if users want to to view the defined confidentiality rules or view a list of tables and plots the confidentiality rules have been applied to.

For data that will not or cannot be aggregated, FishSET provides a set of functions to strip confidential information while retaining the underlying structure of the data (Table 4.9). These functions turn the numeric data into categories using quantiles and within group percentiles, running sum, or lagged difference. To view documentation for these functions, visit the FishSET GitHub webpage.

Purpose	Function
Set confidentiality rules	set_confid_check()
Recode variables based on quantiles	set_quants()
Transform variable to within group percentiles	group_perc()
Transform variable to within group running sum	group_cumsum()
Transform variable to within group lagged difference	group_diff()
Randomize value by percentage range	randomize_value_range()
Randomize value between rows	randomize_value_row()
Randomize latitude and longitude points by zone	randomize_lonlat_zone()
Jitter longitude and latitude points	jitter_lonlat_zone()

Table 4.9 Function list for transforming confidential data.

FishSET GUI

Confidentiality rules are defined on the `Upload Data` tab. After data are loaded, press the blue `Confidentiality` button. Select the *vessel identifier*, *rule*, and *threshold value* for the rule. Select the green `Save` button. Repeat to add another rule. The *n* rule ("rule of n") defaults to 3, at least three unique observational units (e.g., vessels). The *k* rule ("identification of majority allocation") defaults to 90; no single observational unit can account for 90% or more of the value. Once rules are saved, close the window. The confidentiality rules will be applied to output tables and plots. If a confidentiality rule is broken, the value will be suppressed in the output table or plot. The table and plot will also be saved but without data suppression.

All functions listed in Table 4.9 to transform data are found in the `Data Transformations` option on the `Compute New Variables` tab.

5 Metadata

Descriptive information about data, metadata, provides a reference of the attributes and history of the data. Because R data must be in a dataframe, metadata must be stored in a separate file from the data. We recommend taking the time to develop a metadata file for each data file saved in the FishSET database. Metadata can increase the longevity of a data table by ensuring that information, such as units, is not lost over time, improve organization of the data, and improve ability to integrate and share the data.

FishSET provides fillable and editable forms that are saved as .json files in the project-specific data subdirectory in the FishSETFolder directory. The form is identical, whether called in the R console or the FishSET GUI.

5.1 Functions

R console

Open the metadata form by running the `metadata_gui()` function in the R console.

FishSET GUI

The metadata form can be accessed through the Upload Data tab. A project does not have to be specified.

5.2 Filling out the forms

The metadata form opens on the create tab.

Steps:

1. Select a project and a table from the project.
2. If the table does not appear in the *Select Table* box, press the *Refresh tables* button. The metadata created will be associated with the selected table from the selected project. The top part of the form contains boxes for standard information about the data.
3. Press the blue *Load data* button. This will add boxes to provide descriptions for each variable in the data table.
4. When you are done filling out boxes, click the blue *Create meta* button.
5. Press the red *Close* button to exit metadata creation.

6 Exploratory data analysis

Exploratory data analysis is an important preliminary step to gain insight into your data and focus analyses. This process of inspecting your data, especially visually, creates a broad picture of important trends and major points to study in greater detail. The output can be used for exploration and output for reports, such as describing the fleets or changes in catch effort over time.

6.1 Functions

We group exploratory data analysis into 1) basic exploratory analysis - functions exploring the distribution, availability, and variance of data; 2) fleet summaries – functions to view and understand fleets and identify meaningful groupings; and 3) simple analyses - functions to evaluate relationships between variables and identify redundant variables. To view documentation for these functions, visit the FishSET GitHub webpage.

Purpose	Functions
Basic Exploratory Analysis	
Interactive application to remove variables from data table	select_vars()
Add removed variables back into data table	
Interactive application	add_vars_gui()
Non-interactive version	add_vars()
Filter data	
Create filter rules	filter_table()
Apply filter rules	filter_dat()
View distribution, availability, and variance of data	
Hotspots analysis	

Purpose	Functions
Basic Exploratory Analysis	
Kernel density plot	map_kernel()
Getis-Ord statistic	getis_ord_stats()
Moran's I statistic	morans_stats()
View vessel locations on map *Map may contain confidential data	
Static map of haul locations	map_plot()
Interactive map of haul locations or paths with zones	map_viewer()
View aggregated variable by date and fishing zone	spatial_summary()
Assess spatial variance/clumping of grouping variable	spatial_hist()
View distribution of variable over time	temp_plot()
Fleet and group summary	
View number of vessels by fleet and/or group	vessel_count()
Aggregate species catch by time period	species_catch()
Moving average of catch (specify window size and summary statistic)	roll_catch()
View catch variable summarized over weeks	weekly_catch()
View mean weekly catch per unit effort	weekly_effort()
Compare average CPUE and catch total/share of total catch between one or more species	bycatch()
View trip duration	trip_dur_out()
View density plot (kernel density estimate, cumulative distribution function, or empirical cdf) of selected variable	density_plot()
Simple Analyses	
View the correlation coefficient between numeric variables	corr_out()
Assess fitted relationship between two variables	xy_plot()

Table 6.1 List of functions for exploring and visualizing the data.

6.1.1 R console

For all functions, the `dat` input argument refers to the primary dataset. Use quotes if pulling `dat` from the FishSET database. The `project` input argument is the name of the project and used for organization purposes. Plots are generated using the ggplot2 package and saved to the *Output* folder as .png files. Tables are saved to the *Output* folder as .csv files. All outputs are automatically saved to the *Output* folder in the FishSET package directory.

Fleet and group summaries

Summaries may be grouped by defined fleets or other groupings. Each function has a different purpose but they all have a set of optional parameters that allow for grouping the data (`group`) or filtering by date (`filter_date`) or another variable (`filter_value`). These options default to NULL and do not have to be specified. There are additional options to adjust the appearance of the output. Descriptions of these options are provided below. To view function documentation, visit the FishSET GitHub webpage.

6.1.2 FishSET GUI

In the FishSET GUI, all data exploration functions are located in the `Data Exploration`, `Fleet Summary`, `Map Viewer`, and `Simple Analyses` tabs.

Data Exploration tab

The `Data Exploration` tab defaults to displaying the first 15 lines of the primary data set. Use the `Select a dataset` dropdown option to view port, auxiliary, or gridded data. The displayed table is used to remove variables, edit cells, and filter the data. To remove variables, single click on any cell in the variables and then press the `Remove variables from data set` button. Multiple variables can be removed at once. To edit a cell, double click on the cell. Use the boxes between the column headings and above the first line of data to define how data should be filtered. Choose a number range to retain for numeric data. For categorical data, select the category to retain. The filter definitions will be saved in a table format when the `Save data to FishSET database` button is pushed.

Now select plots from the `View data table or plots` dropdown box in the left panel. Use the `Select Plot Type` dropdown box to switch between *temporal*, *spatial*, *spatial-zone summary,* and *x-y* plots. An estimate of linear relationships can be calculated in the `Simple Analyses` tab. Spatial plots include observed locations and spatial kernels. Users can identify the latitude and longitude of individual points and zoom in. A spatial data set must be provided to view the spatial plots. To view the *Getis Ord* and *Moran's I* statistics, population and additional options below the `Select Plot Type` drop down box.

Fleet Assignment and Summary tab

The `Fleet Assignment and Summary` tab is divided into `Fleet Assignment` and `Fleet Summary` (Figure 6.1) subtabs. Use the `Fleet Assignment` subtab to first define fleets (instructions are provided in the GUI) and then apply the definition to the data. If fleets are not assigned then the data will be treated as a fleet as a whole in the *Fleet Summary* functions.

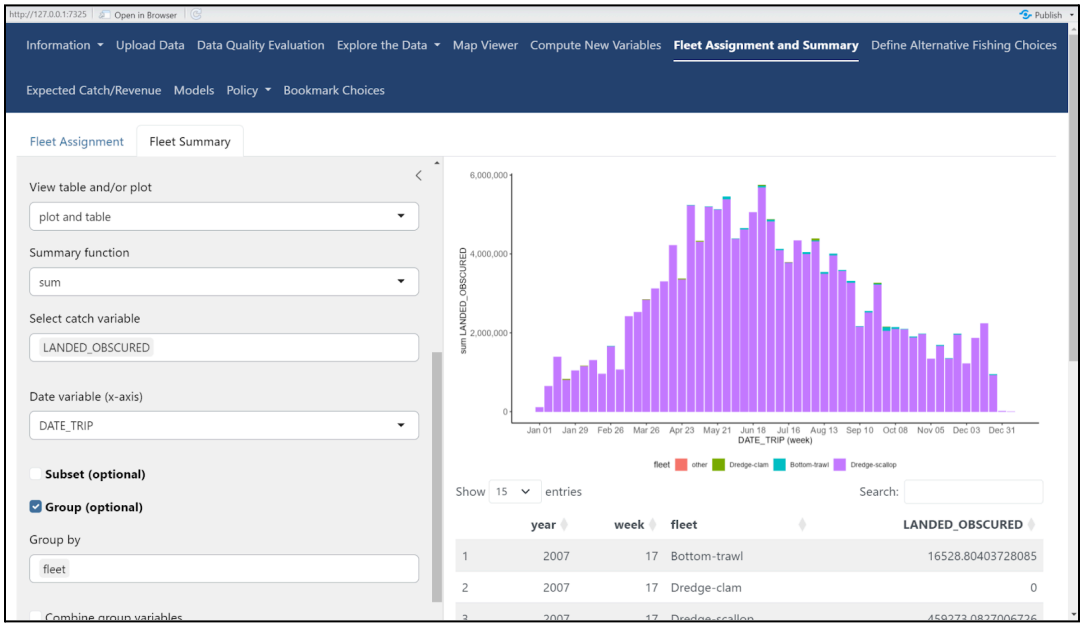


Figure 6.1 View fleet summary plots and tables such as weekly catch by selecting the fleet summary subtab.

Map Viewer tab

The map viewer plot shows vessel haul locations or paths color coded by a numeric variable (Figure 6.2). To view values of individual points, hover over a point on the map. The information will show in the bottom left-hand corner of the plot. Map viewer also requires a zone identifier variable in the primary dataset, which links the primary dataset to the spatial dataset. Go to the `Create New Variables` tab to assign observations to zones if a zonal identifier does not exist in the primary dataset

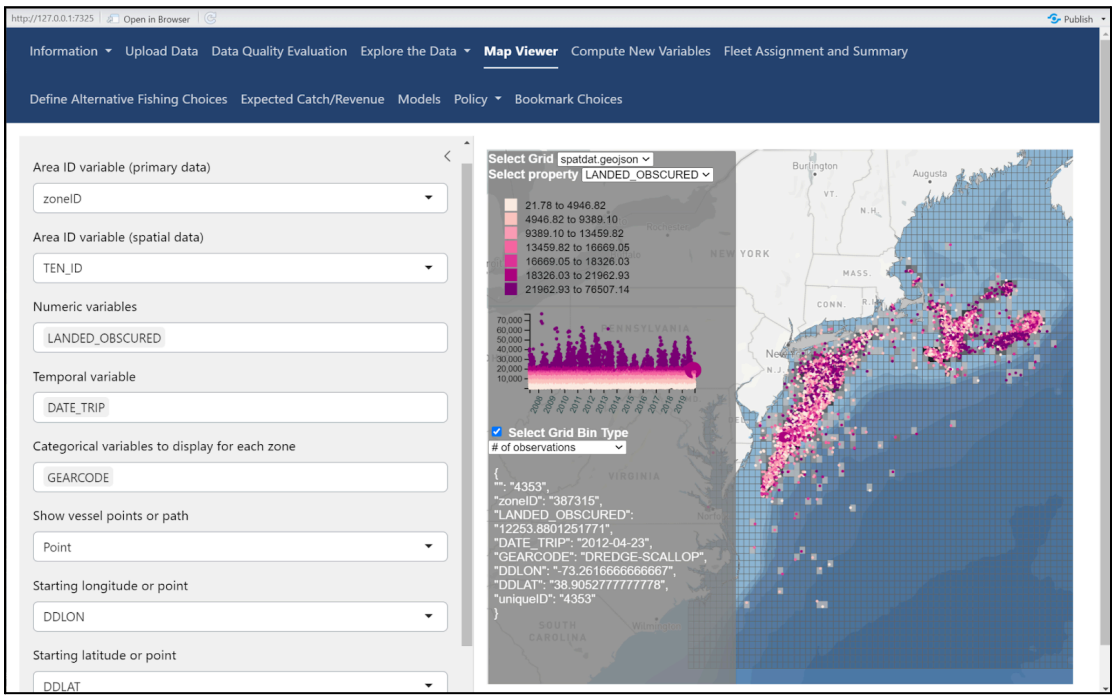


Figure 6.2 Map Viewer output. Points are color coded by the value of the numeric variable LANDED_OBSCURED. Gray shaded areas indicate the number of observations per zone.

Input options:

`Area ID variable (primary data)`: Variable in primary data containing the zone identifier such as ZoneID.

`Area ID variable (spatial data)`: Name of the property in the spatial data file that identifies the zones. Links to Area variable (data).

`Numeric variables`: Numeric variables to include in plotting. This variable is required. Points on the map are color coded based on the numeric variable value. Multiple variables can be chosen but only one variable will be plotted at a time.

`Temporal variables`: Temporal variables to plot the numeric variable against. The scatter plot is provided on the left-hand side of the map. Multiple temporal variables can be included but only one will be plotted at a time.

`Categorical variables to display on map` (optional): Categorical variables to display on map. Multiple categorical variables can be included but only one will be plotted at a time.

`Location point or path`: Should the map show hauls as individual points or paths?

If points are to be plotted

Longitude point

Latitude point
If path is to be plotted
Starting longitude
Starting latitude
Ending longitude
Ending latitude

Simple Analyses tab

Relationships between variables can be viewed in the `Simple Analyses` tab. FishSET defaults to showing a matrix and table of correlation values (Pearson's correlation coefficient) between numeric variables (Figure 6.3). Use the box above the table to remove or add variables from the matrix. Note that this will not remove variables from the dataset, only from the correlation matrix and table.

To show simple linear regressions, select `Regression` from the `Show Correlation or Simple Linear Regression` drop down box in the left-hand panel (Figure 6.4). The summary function output, including coefficients and residuals, are automatically displayed when response or explanatory variables are changed. The summary output can be saved to the *Output* folder by pressing the `Save table to folder as csv` button. Two plots are also produced. The first shows the best fit line for the two variables. The gray standard error bounds are computed using the `predict()` function. The second plot shows the residuals against fitted values. Plots can be saved to the *Output* folder by pressing the `Save plots to folder` button.

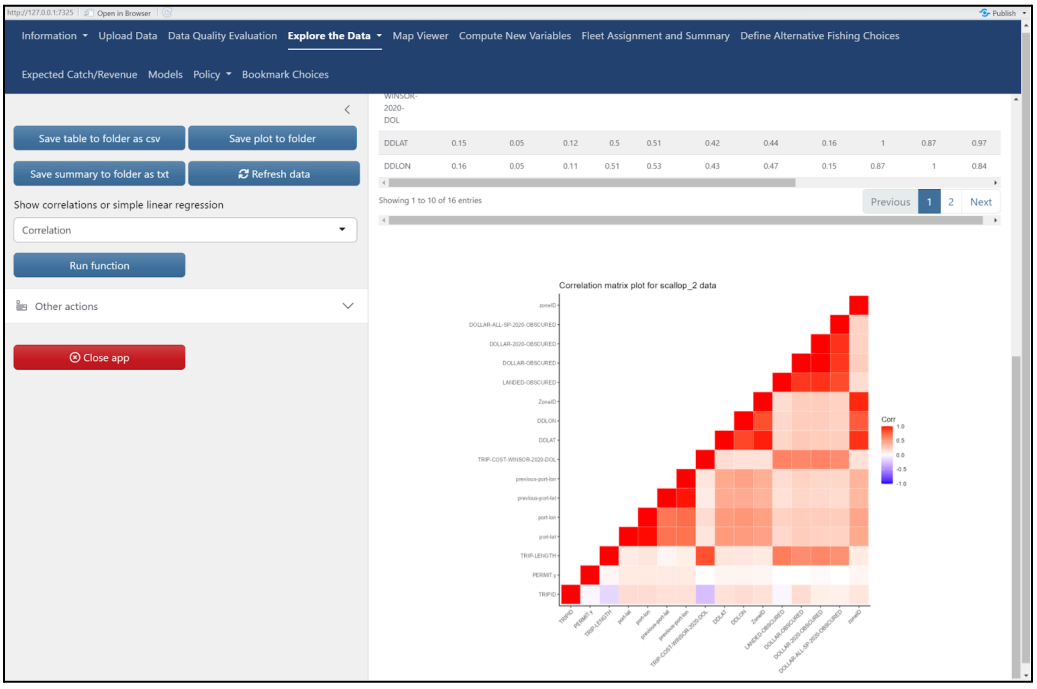


Figure 6.3 Correlation table and matrix plot. Variables can be removed or added in the Select Variables box.

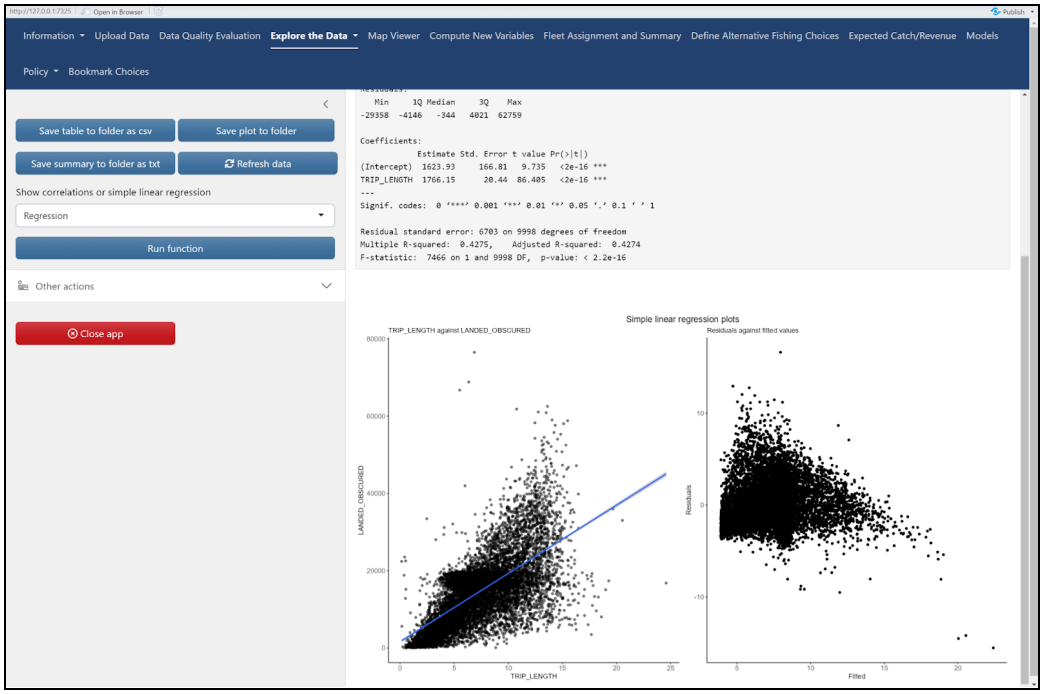


Figure 6.4 Simple linear regression output and plots.

7 Transforming and Creating Data

FishSET provides a number of functions for modifying and creating data. Functions are divided into arithmetic, data transformations, dummy, nominal ID, spatial, temporal, and trip-level functions. Arithmetic functions include catch per unit effort and undirected arithmetic functions. Data transformation functions focus on transforming numeric data into categories. Dummy functions create variables that allow for comparing and contrasting, such as pre- vs. post- a management action. The nominal ID functions convert a variable into nominal data to identify unique hauls or trips and fishery seasons. Spatial functions focus on transforming data to measures of distance and midpoints. Temporal functions are used to extract the desired unit of time and duration of time. Trip-level functions are used to collapse data from haul to trip level and calculate trip distance and trip centroid. To view documentation for these functions, visit the FishSET GitHub webpage.

7.1 Functions

Purpose	Function
Arithmetic	
Numeric functions	create_var_num()
Catch per unit effort	cpue()
Data transformations	
Within-group percentage	group_perc()
Within-group lagged difference	group_diff()
Within-group running sum	group_cumsum()
Quartiles	set_quants()
Dummy	

*The different outputs for dummy variable functions are defined with 'value' and 'opts' parameters	
From variable	dummy_num()
From policy dates	dummy_num()
From area/zone closures	dummy_num()
Nominal ID	
Haul or trip identifier	ID_var()
Binary Fisher season identifier	seasonalID()
Fishery season identifier based on location, species, gear	create_seasonal_ID()
Spatial	
Assign observations to zones	assignment_column()
Distance between two points	create_dist_between()
Midpoint location for each haul (lat/lon)	create_mid_haul()
Zone when choice of where to go next was made	create_startingloc()
Temporal	
Convert or extract time unit	temporal_mod()
Duration of time between two temporal variables	create_duration()
Trip-level	
Collapse data from haul-level to trip-level	haul_to_trip()
Calculate trip distance	create_distance()
Calculate trip centroid	create_trip_centroid()

Table 7.1 List of functions for modifying and creating variables.

7.2 R console

For all functions listed in table 7.1, the input argument `dat` refers to the name of the primary dataset. Use quotes if the dataset is being pulled from the FishSET database. The input argument `name` (optional) is the name of the new variable to be created. The `name` defaults to the function name if not specified. To view documentation for these functions and a description of additional input arguments, visit the FishSET GitHub webpage.

7.3 FishSET GUI

All data transformation and creation functions are provided in the `Compute New Variable` tab in the FishSET GUI. First, select the function category from the `Create variables based on` the dropdown box. Then select the function to run from the `Functions` dropdown box. The `Name of new variable` will default to the name of the function if it is not filled in. Click the green `Run Function` button to create the variable. The new variable will be displayed at the end of the table in the main panel. Before moving on to model development, click the `Save data to the FishSET database` button.

8 Model Development

8.1 Overview

Model development in FishSET requires preparing the data, choosing a likelihood function, and specifying initial parameter values. This section provides details on the required and optional data, model choices, and model outputs, which rely on the functions in table 8.1. To view documentation for these functions, visit the FishSET GitHub webpage.

Function	Purpose
`create_startingloc()`	Create a variable containing the starting location/zone
`previous_loc()`	Create a variable of the previous location/zone/port
`assignment_column()`	Assign zone IDs from spatial data to observations in the primary dataset
`create_alternative_choice()`	Define alternative fishing choices
`create_expectations()`	Define expected catch/revenue matrix
`make_model_design()`	Make model design
`discretefish_subroutine()`	Run model
`globalcheck_view()`	Model Error
`model_out_view()`	Model Output
`model_params()`	Model parameters
`model_fit()`	Model Comparison Metrics
`select_model()`	Select best model

Table 8.1 Model development functions included in FishSET.

8.2 Data

All models require the chosen fishing location (and corresponding zone ID), location when choice of where to fish was made (e.g., a port or zone ID), and catch data at that location be included in the primary dataset. Also, all models require a distance matrix to be generated. Price data and expected catch matrices are required for some models. Additional optional data that interact with travel distance or vary by location may also be added.

8.2.1 Starting location

Location choice models in FishSET require a starting location variable that represents the location prior to the fishing occasion. Generally, the starting location for a fishing trip is the port of departure. For set- or haul-level data the starting location could be the zone ID for the previous set/haul. Essentially, the starting location vector is the location of a vessel when the decision of where to fish next was made. IMPORTANT NOTE: if the primary dataset is at the trip level and port information is included in the data, then users can move forward to model design without the following step. However, when working with set/haul level data and the previous location is not included in the primary dataset, use the `create_startingloc()` function to generate a vector with starting locations prior to model design. Visit the FishSET GitHub webpage to view documentation for this function.

To create the starting location in the FishSET GUI, select the `Compute New Variables` tab. Choose `"Spatial functions"` in the `"Create Variables based on"` dropdown box. Select `"Zone when choice of where to go next was made"` from the `Functions` dropdown menu, and fill in the input options.

8.2.2 Distance matrix

A distance matrix, defined as the distance between the starting location and all alternative fishing location choices, is created using the `create_alternative_choice()` function (which defines the alternatives) and then called by the `make_model_design()` function (which creates the distance matrix). The distance matrix is saved in the model design file in the FishSET database and pulled when the model run function is called. Visit the FishSET GitHub webpage to view documentation for these functions.

The distance matrix can be generated from the primary data (`dat`) or from gridded data (`griddedDat`).

Generate the distance matrix from `dat`

If the distance matrix is calculated from the primary haul-level data (`dat`), the function requires defining how the longitude and latitude for starting location (`occasion`) and alternative locations (`alt_var`) should be found.

Choices for the starting location (`occasion`) are the centroid of the zone where the most recent previous haul occurred (`occasion` = "centroid"), port variable in `dat` (for example, `occasion` = 'disembarked_port'), or paired latitude and longitude variables for the most recent previous haul in `dat` such as haul starting location (`occasion` = c('lon_start', 'lat_start').

Choices for alternative fishing locations (`alt_var`) are the zonal centroid of alternative fishing zones (`alt_var` = "zonal centroid"), centroid of fishing (`alt_var` = "fishing centroid"), or nearest point to alternative zones (`alt_var` = "nearest point").

a. *If* `alt_var` *or* `occasion_var` *are centroids:*
   A table of the centroids for each zone/area must be created using the `create_centroid()` function prior to defining alternative fishing choices. If zonal centroids have been previously calculated and a centroid table exists in the FishSET database, then `spat` can be NULL. The function will call the centroid table from the FishSET database. Go to the FishSET website to view details on how to use the `create_centroid()` function.

b. *If* `occasion_var` *is a port variable:*
   A port table containing the latitude and longitude of ports is required. If a port table does not exist in the FishSET database, run the `load_port()` function. The starting location will be the latitude and longitude of the port.

c. *If* `alt_var` *or* `occasion_var` *are represented by longitude and latitude variables:*
   Use longitude and latitude variables from primary data and specified as c(lon, lat).

The distance matrix is then calculated between defined starting locations and alternative choice locations. Distance (`dist.units`) can be represented in miles, kilometers, or meters. The `min.haul` parameter specifies the minimum number of hauls that must be recorded in a zone to be included in analysis. Use this argument to remove data from zones with insufficient data from analyses.

Generate the distance matrix from `griddedDat`

If using a gridded dataset, only `dat`, `project`, and `griddedDat` must be specified. Columns in the gridded data file must be individual zones and must match the zone identifier variable in the primary dataset. The gridded data can vary over a second dimension, such as date, but must match a variable in `dat`. See Chapter 4.1.4 for more information. Save the gridded data file to the FishSET database using the `load_grid()` function.

Generate the distance matrix in the FishSET GUI

Go to the `Create New Variables` tab. In the `Create New Variables Based on` dropdown menu, select the "`Spatial functions`" option. In the `Functions` dropdown menu, select the

"Zonal centroid" or "Fishery centroid" option if using centroids. Then go to the Define Alternative Fishing Choices tab, define input options and save choices.

8.2.3 Expected catch and revenue matrices

The Expected catch matrix contains expected catch for the full set of alternatives. The matrix is calculated as a moving average of catch over time. The user determines the time window for which this average is calculated and the lag in days for the time window. This matrix is required to run the conditional logit model that utilizes an expected catch specification.

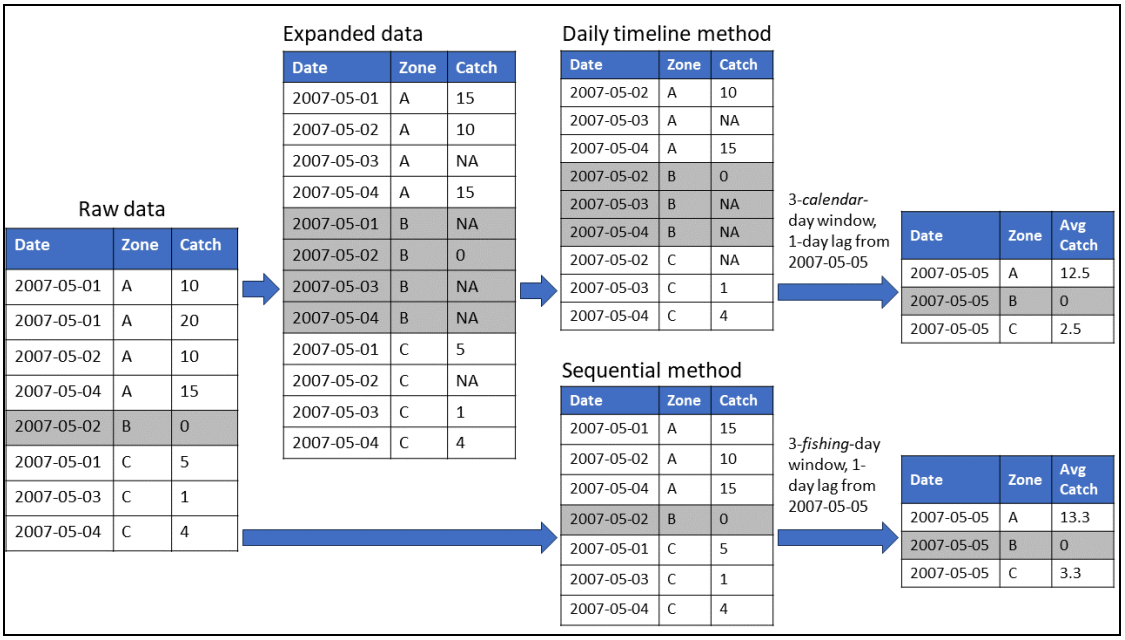
The create_expectations() function will generate the expected catch matrix and requires dat, project name, and a catch variable. Expected catch can be calculated using historical data of catch in each zone across the entire dataset (defineGroup = "fleet") or within groups using the defineGroup option.

The create_expectations() function can also be used to generate expected revenue by including a price or value variable from dat. The value entered for price is multiplied by catch to generate revenue. If revenue exists in dat and you wish to use this revenue instead of price, then set catch equal to the revenue variable and leave the price input empty. There are a number of choices defining whether expected catch should come from observed catch averaged over short or long time windows.

1. Identify the temporal variable (temp.var input for create_expectations) in dat to use. The default window (temp.window input for create_expectations) is set to the most recent seven observations.
2. Select whether to use a sequential (temporal = "sequential") or daily timeline (temporal = "daily"). Sequential timeline sorts data by date but does not take into account that catch or revenue may be missing for some dates (Figure 8.1). Daily timeline adds in dates with NAs or missing values (Figure 8.1). With the sequential timeline, a window size of seven means catch would be the average over the past seven days when *fishing* actually occurred in that zone. With the daily timeline, it would be the average over seven *calendar* days. Figure 8.1 below describes the difference in these two methods.

The standard average catch method (calc.method = "standardAverage") uses the specified moving window parameters to create a matrix of average catch with zone*group creating rows and date the columns. Alternatively, the simple lag regression of the mean (calc.method = "simpleLag") calculates the predicted value for each zone*group and date given regression coefficients p for each zone*group. The lag method (lag.method) can use regression over the entire dataset ("simple") or for grouped time periods ("grouped").

The expected catch matrix is pulled from the matrix of calculated catches for each date and zone. The matrix is of dimensions [(number of rows in dat)*(number of alternatives)]. Expected catch is filled out by mapping the calculated catch for each zone given the observed date and group in dat.



Raw data

Date	Zone	Catch
2007-05-01	A	10
2007-05-01	A	20
2007-05-02	A	10
2007-05-04	A	15
2007-05-02	B	0
2007-05-01	C	5
2007-05-03	C	1
2007-05-04	C	4

Expanded data

Date	Zone	Catch
2007-05-01	A	15
2007-05-02	A	10
2007-05-03	A	NA
2007-05-04	A	15
2007-05-01	B	NA
2007-05-02	B	0
2007-05-03	B	NA
2007-05-04	B	NA
2007-05-01	C	5
2007-05-02	C	NA
2007-05-03	C	1
2007-05-04	C	4

Daily timeline method

Date	Zone	Catch
2007-05-02	A	10
2007-05-03	A	NA
2007-05-04	A	15
2007-05-02	B	0
2007-05-03	B	NA
2007-05-04	B	NA
2007-05-02	C	NA
2007-05-03	C	1
2007-05-04	C	4

3-*calendar*-day window, 1-day lag from 2007-05-05

Date	Zone	Avg Catch
2007-05-05	A	12.5
2007-05-05	B	0
2007-05-05	C	2.5

Sequential method

Date	Zone	Catch
2007-05-01	A	15
2007-05-02	A	10
2007-05-04	A	15
2007-05-02	B	0
2007-05-01	C	5
2007-05-03	C	1
2007-05-04	C	4

3-*fishing*-day window, 1-day lag from 2007-05-05

Date	Zone	Avg Catch
2007-05-05	A	13.3
2007-05-05	B	0
2007-05-05	C	3.3

Figure 8.1 Example of daily timeline and sequential averaging methods. Note that 'NA' values are filled in for missing date-zone combinations in the daily timeline method, but these values are omitted when calculating averages.

FishSET offers tools to evaluate and handle sparse data. The data used to generate the expected catch matrix may be "sparse" if there are zero observations in a particular zone and time window. Data sparsity may create convergence issues or be inconsistent with a user's assumptions about the underlying behavior. How to handle sparse data must be carefully considered at this step. Empty expected catch values can be filled in based on a user's assumption, but doing so can lead to biased or misleading results.

Sparsity can be evaluated with the `temp_obs_table()` function. If there are many empty values, consider changing the temporal arguments to reduce data sparsity. A longer time window may reduce sparsity, but may require increasingly tenuous assumptions. The `empty.expectations` input can be set to 0 or 1e-04. These options should be used cautiously and tested for robustness. See Chapter 11 Tips for more information on sparsity in datasets.

By default, the `create_expectations()` function does not include rows with missing (NA values) catch, revenue, or price in the calculation of the moving averages. The `empty.catch` input can be set to NA, 0, the average of all catch values (`allCatch`), or the average of grouped catch values (`groupedCatch`). Values of 0 in the catch variable are considered times when fishing activity occurred but nothing was caught. These zero values *are* included in rolling average calculations.

8.2.4 Travel-distance and grid-varying variables

Other data used in models are travel-distance and grid-varying variables. These explanatory variables are optional. Examples include an externally-calculated expected catch matrix, vessel characteristics, or wind speed within zones.

8.3 Likelihood functions

FishSET currently provides six statistical functions for maximum likelihood estimation. Users should consult the references provided (Table 8.2) to understand the assumptions of each likelihood and determine if it is appropriate for their data and question.

FishSET function	Statistical function	Reference
logit_c	Conditional logit	McFadden 1974
logit_zonal	Zonal (area-specific constants) logit	Abbot and Wilen 2011
epm_normal	Expected profit model with normal catch function	Haynie and Layton 2010
epm_weibull	Expected profit model with Weibull catch function	Haynie and Layton 2010
epm_lognormal	Expected profit model with lognormal catch function	Haynie and Layton 2010
logit_correction	Full information model with Dahl's correction function	Dahl 2002

Table 8.2 Statistical functions included in FishSET and references for each function.

8.3.1 Conditional and zonal logit (logit_c and logit_zonal)

For the conditional logit model, the probability of fishing a location for a given observation is:

$$p_{i,j} = \frac{exp\left(\sum_{m=1}^{M} \beta_m G_{i,j,m} + \sum_{n=1}^{N} \gamma_n T_{i,n} D_{i,j}\right)}{\sum_{k=1}^{K} exp\left(\sum_{m=1}^{M} \beta_m G_{i,k,m} + \sum_{n=1}^{N} \gamma_n T_{i,n} D_{i,k}\right)},$$

where $p_{i,j}$ is the probability of selecting alternative j for choice occasion i, K is the total number of alternatives, and M is the total number of grid-varying variables.

Travel-distance variables ($T_{i,n}$; where n = 1, 2, …, N) are alternative-invariant variables that are interacted with travel distances for each alternative ($D_{i,k}$) to form the cost portion of the likelihood. For example, vessel characteristics that affect how much disutility is suffered by traveling a greater distance. Each $T_{i,n}$ variable name corresponds to data with dimensions (number of observations) x (1) and returns a single parameter. For example, for observation i and alternative-invariant variable $n = 1$, the cost portion of the logit model for each alternative where $K = 4$ is:

Location 1	Location 2	Location 3	Location 4
$\gamma_1 T_{i,1} D_{i,1}$	$\gamma_1 T_{i,1} D_{i,2}$	$\gamma_1 T_{i,1} D_{i,3}$	$\gamma_1 T_{i,1} D_{i,4}$

Grid-varying variables ($G_{i,k,m}$) are alternative-specific variables, such as catch rates or the expected catch matrix. Each $G_{i,k,m}$ variable name therefore corresponds to data with dimensions (number of observations) x (K). For example, for observation i and $m = 1$, the value for each alternative where $K = 4$ is:

Location 1	Location 2	Location 3	Location 4

$\beta_1 G_{i,1,1}$	$\beta_1 G_{i,2,1}$	$\beta_1 G_{i,3,1}$	$\beta_1 G_{i,4,1}$

Conditional and zonal logits are versions of the same model. However, the probability of fishing a location for a given observation for zonal logit is:

$$p_{ij} = \frac{exp\left(\sum_{m=1}^{M} \beta_{jm} G_{im} + \sum_{n=1}^{N} \gamma_n T_{in} D_{ij}\right)}{\sum_{k=1}^{K} exp\left(\sum_{m=1}^{M} \beta_{km} G_{im} + \sum_{n=1}^{N} \gamma_n T_{in} D_{ik}\right)}.$$

The travel-distance variables ($T_{i,n}$) are the same as in the conditional logit model. The difference is that the $G_{i,m}$ variables do not vary across alternatives for the zonal logit model and each variable corresponds to data with dimensions (number of observations) x (1). For example, $G_{i,m}$ might represent vessel gross tonnage, thus β_j parameters capture the effect of changes vessel gross tonnage on the utility of zone *j*. The model returns β_j parameters for K - 1 alternatives because only differences in these values matter, and the value of $\beta_{j=1}$ is normalized to zero.

Location 1	Location 2	Location 3	Location 4
$(\beta_1 = 0)G_{i,1}$	$\beta_2 G_{i,1}$	$\beta_3 G_{i,1}$	$\beta_4 G_{i,1}$

For this likelihood, the parameter order is: c[(alternative-specific parameters), (travel-distance parameters)]. For the conditional logit, the length of the parameter vector is the number of alternative-specific variables plus the number of travel-distance variables. For the zonal logit, the length is the number of alternative-specific variables multiplied by the number of alternatives, plus the number of travel-distance variables.

8.3.2 Expected profit model with normal catch function (epm_normal)

$$\ell_{ij} = \underbrace{\frac{1}{\sigma_j\sqrt{2\pi}} exp\left(-\frac{\left[Y_{ij} - \Sigma_m^M \beta_{jm} G_{im}\right]^2}{2\sigma_j^2}\right)}_{1} \times \underbrace{\left(\frac{exp\left[\frac{P_i\left(\Sigma_m^M \beta_{jm} G_{im}\right) + \Sigma_n^N \gamma_n T_{in} D_{ij}}{\sigma_\varepsilon}\right]}{\Sigma_a^A exp\left[\frac{P_i\left(\Sigma_m^M \beta_{am} G_{im}\right) + \Sigma_n^N \gamma_n T_{in} D_{ia}}{\sigma_\varepsilon}\right]}\right)}_{2}$$

The expected profit model (EPM) with a normal catch function can be useful for preliminary analyses and data exploration, but the normal catch function supports positive and negative values of catch. Thus, EPM models with alternative catch functions should be considered for formal analyses.

Part 1 of the equation is the normal catch function of the likelihood, and part 2 is the logit choice component (following the structure of the EPM Weibull model presented in Haynie and Layton 2010). The mean of the normal function, which also represents the expected mean catch in the logit choice component, is parameterized as:

$\Sigma_m^M \beta_{jm} G_{im}$,

where i is a choice occasion, j is a fishing location, β_{jm} are alternative-specific parameters that interact with choice occasion variables G_{im}, and users can choose M number of variables (where m = 1, 2, …, M). The G_{im} variables do not vary across alternatives (e.g., vessel gross tonnage), but can affect catch differently across alternatives through alternative-specific β_{jm} parameters. Each G_{im} variable name corresponds to data with dimensions (number of observations) * (1), and returns A parameters where A equals the number of alternatives. The standard deviation (σ_j) of the normal function is constrained to positive values via an exponential function, and Y_{ij} is the actual catch. By default, only a single standard deviation value is specified, but alternative-specific values can be specified as well. In the logit choice component of the model, the expected mean catch is multiplied by the price (P_i) to calculate expected revenues. The cost portion of the likelihood is specified as:

$$\Sigma_n^N \gamma_n T_{in} D_{ij},$$

where γ_n are alternative-invariant parameters that interact with travel distance D_{ij} and alternative-invariant variables T_{in}, and users can choose N number of variables (where n = 1, 2, …, N). Each T_{in} variable name corresponds to data with dimensions (number of observations) * (1), and may represent spatial, vessel, market, and/or port characteristics (Haynie and Layton 2010). A single γ_n parameter is returned for each T_{in} variable. The common scale parameter σ_ε is distributed as Type I Extreme Value and represents unobserved expected profit heterogeneity (Haynie and Layton 2010).

For this likelihood, the parameter order is: c[(alternative-specific parameters), (travel-distance parameters), (standard deviation), (common scale parameter)]. The length of the parameter vector is the number of alternative specific variables multiplied by the number of alternatives, plus the number of travel distance variables, plus standard deviation parameters (a single value can be specified or alternative-specific standard deviation can be used) and the common scale parameter.

8.3.3 Expected profit model with Weibull catch function (epm_weibull)

$$\ell_{ij} = \underbrace{\left(\frac{k}{\Sigma_m^M \beta_{jm} G_{im}} \left[\frac{Y_{ij}}{\Sigma_m^M \beta_{jm} G_{im}} \right]^{k-1} exp\left[- \left(\frac{Y_{ij}}{\Sigma_m^M \beta_{jm} G_{im}} \right)^k \right] \right)}_{1} \times \underbrace{\left(\frac{exp\left[\frac{P_i \left(\Sigma_m^M \beta_{jm} G_{im} \right) \Gamma(1+1/k) + \Sigma_n^N \gamma_n T_{in} D_{ij}}{\sigma_\varepsilon} \right]}{\Sigma_a^A exp\left[\frac{P_i \left(\Sigma_m^M \beta_{am} G_{im} \right) \Gamma(1+1/k) + \Sigma_n^N \gamma_n T_{in} D_{ia}}{\sigma_\varepsilon} \right]} \right)}_{2}$$

Part 1 of the equation is the Weibull catch function of the likelihood, and part 2 is the logit choice component (Haynie and Layton 2010). The scale portion of the Weibull function is parameterized as:

$$\Sigma_m^M \beta_{jm} G_{im},$$

where i is a choice occasion, j is a fishing location, β_{jm} are alternative-specific parameters that interact with choice occasion variables G_{im}, and users can choose M number of variables (where m = 1, 2, …, M). The G_{im} variables do not vary across alternatives (e.g., vessel gross tonnage), but can affect catch differently across alternatives through alternative-specific β_{jm} parameters. Each G_{im} variable name corresponds to data with dimensions (number of observations) * (1), and returns A parameters where A equals the number of alternatives. The scale portion and shape parameter k of the Weibull function are

constrained to values greater than 0 via exponential functions, and Y_{ij} is the actual catch (Haynie and Layton 2010). The expected mean catch function (i.e., mean of the Weibull distribution) is parameterized in the logit choice component of the likelihood as:

$$\left(\Sigma_m^M \beta_{jm} G_{im}\right)\Gamma(1 + 1/k),$$

and multiplied by price P_i to calculate expected revenues. The cost portion of the likelihood is specified as:

$$\Sigma_n^N \gamma_n T_{in} D_{ij},$$

where γ_n are alternative-invariant parameters that interact with travel distance D_{ij} and alternative-invariant variables T_{in}, and users can choose N number of variables (where n = 1, 2, …, N). Each T_{in} variable name corresponds to data with dimensions (number of observations) * (1), and may represent spatial, vessel, market, and/or port characteristics (Haynie and Layton 2010). A single γ_n parameter is returned for each T_{in} variable. The common scale parameter σ_ε is distributed as Type I Extreme Value and represents unobserved expected profit heterogeneity (Haynie and Layton 2010).

An optional extension for the `epm_weibull` likelihood function is to estimate alternative-specific shape parameters k_j.

For this likelihood, the parameter order is: c[(alternative-specific parameters), (travel-distance parameters), (shape parameters), (common scale parameter)]. The length of the parameter vector is the number of alternative specific variables multiplied by the number of alternatives, plus the number of travel distance variables, plus shape parameters (a single value can be specified or alternative-specific shape values can be used) and the common scale parameter.

8.3.4 Expected profit model with lognormal catch function (epm_lognormal)

$$\ell_{ij} = \underbrace{\left(2\pi\sigma_j^2 Y_{ij}^2\right)^{-\frac{1}{2}} exp\left(\frac{-\left[ln\left(Y_{ij}\right) - \Sigma_m^M \beta_{jm} G_{im}\right]^2}{2\sigma_j^2}\right)}_{1} \times \underbrace{\left(\frac{exp\left[\frac{P_i exp\left(\Sigma_m^M \beta_{jm} G_{im} + \frac{\sigma_j^2}{2}\right) + \Sigma_n^N \gamma_n T_{in} D_{ij}}{\sigma_\varepsilon}\right]}{\Sigma_a^A exp\left[\frac{P_i exp\left(\Sigma_m^M \beta_{am} G_{im} + \frac{\sigma_a^2}{2}\right) + \Sigma_n^N \gamma_n T_{in} D_{ia}}{\sigma_\varepsilon}\right]}\right)}_{2}$$

The overall structure of the function follows the expected profit model with the Weibull catch function presented in Haynie and Layton (2010). Part 1 of the equation is the log-normal catch function of the likelihood, and part 2 is the logit choice component. The expected (or mean) value in the log-normal probability density function is parameterized as:

$\Sigma_m^M \beta_{jm} G_{im}$,

where i is a choice occasion, j is a fishing location, β_{jm} are alternative-specific parameters that interact with choice occasion variables G_{im}, and users can choose M number of variables (where m = 1, 2, …, M). The G_{im} variables do not vary across alternatives (e.g., vessel gross tonnage), but can affect catch differently across alternatives through alternative-specific β_{jm} parameters. Each G_{im} variable name corresponds to data with dimensions (number of observations) * (1), and returns A parameters where A equals the number of alternatives. The standard deviation (σ_j) of the log-normal function is constrained to positive values via an exponential function, and Y_{ij} is the actual catch. By default, only a single standard deviation value is specified, but alternative-specific values can be specified as well. The expected mean catch function (i.e., mean of the log-normal distribution) is parameterized in the logit choice component of the likelihood as:

$$exp\left(\Sigma_m^M \beta_{jm} G_{im} + \frac{\sigma_j^2}{2}\right),$$

and multiplied by price P_i to calculate expected revenues. The cost portion of the likelihood is the same as the expected profit model with Weibull catch function above (see sections 8.4.3 or 8.4.4). Note that the standard deviation of the log-normal catch function (σ_j) is not the same as the common scale parameter σ_ε, which is distributed as Type I Extreme Value and represents unobserved expected profit heterogeneity (Haynie and Layton 2010).

For this likelihood, the parameter order is: c[(alternative-specific parameters), (travel-distance parameters), (standard deviation parameters), (common scale parameter)]. The length of the parameter vector is the number of alternative specific variables multiplied by the number of alternatives, plus the number of travel distance variables, plus standard deviation parameters (a single value can be specified or alternative-specific standard deviations can be used) and the common scale parameter.

8.3.5 Full information model with Dahl's correction function (logit_correction)

$$l_{ij} = \frac{1}{\left(2\pi\sigma_j^2\right)^{1/2}} exp\left[\frac{-\left(y_i - \sum_{m=1}^{M}\left(\beta_{jm} G_{im}\right) - \eta\left(\beta_{jsd}\right)\right)^2}{2\sigma_j^2}\right] \frac{exp\left[\alpha\left(\sum_{m=1}^{M}\left(\beta_{jm} G_{im}\right) + \eta\left(\beta_{jsd}\right)\right) + \sum_{n=1}^{N} \gamma_n T_{in} D_{ij}\right]}{\sum_{k=1}^{K} exp\left[\alpha\left(\sum_{m=1}^{M}\left(\beta_{km} G_{im}\right) + \eta\left(\beta_{ksd}\right)\right) + \sum_{n=1}^{N} \gamma_n T_{in} D_{ik}\right]}$$

Travel-distance variables (`T_in`) are alternative-invariant variables that are interacted with travel distance to form the cost portion of the likelihood. Each variable name corresponds to data with dimensions (number of observations) * (1), and returns a single parameter. They do not interact with alternatives.

$\gamma_1 T_{i1} D_{i1}$	$\gamma_1 T_{i1} D_{i2}$
$\gamma_1 T_{i1} D_{i3}$	$\gamma_1 T_{i1} D_{i4}$

Grid-varying variables are catch-function variables (`G_im`) that are interacted with zonal constants to form the catch portion of the likelihood, such as variables that affect catches across locations. They do not vary across alternatives. Each variable name corresponds to data with dimensions (number of observations) * (1), and returns (`k`) parameters where (`k`) equals the number of alternatives. Note in this

likelihood the catch-function variables vary across observations but not for each location: they are allowed to affect catches across locations due to the location-specific coefficients. Users can choose `M` variables, but for the variable `m=1`, there is only one column for all alternatives.

$\beta_{11}G_{i1}$	$\beta_{\gamma_1}G_{i1}$
$\beta_{\alpha_1}G_{i1}$	$\beta_{A1}G_{i1}$

The variable `startloc` is a required vector which corresponds to the starting location when the agent decides between alternatives. It is created with the `create_startingloc()` function.

The variable `polyn` is the chosen polynomial degree (`d`). Expected catches are corrected with a polynomial approximation of the conditional effort term, $\eta\left(\beta_{ksd}\right)$. Individual approximations are used depending on if the agent "stayed" (`s=1`) or "moved" (`s=0`), and for each location `k`. For each approximation, a coefficient is estimated for each polynomial degree (`d`) as well as two additional interaction terms if the agent 'moved' (`s=0`). See ???? for additional information.

$\eta\left(\beta_{1sd}\right)$	$\eta(\beta_{2sd})$
$\eta(\beta_{\alpha sd})$	$\eta(\beta_{Asd})$

For this likelihood, the parameter order is: c[(marginal utility from catch), (catch-function parameters), (polynomial starting parameters), (travel-distance parameters), (catch sigma)]. The marginal utility from catch and catch sigma are of length equal to unity respectively. The catch-function and travel-distance parameters are of length (number of catch variables) * (`k`) and number of cost variables respectively. The number of polynomial interaction terms is currently set to 2, so given the chosen degree `polyn` there should be `(((polyn+1)*2)+2)*(k)` polynomial starting parameters, where `k` equals the number of alternative fishing choices.

8.4 Initial parameters

The number of initial parameter values required depends upon the number of optional variables included and the chosen model. Model-specific details on the number and order or starting parameter values are provided in the previous section, Likelihood functions. The FishSET GUI automatically calculates the number of initial parameter values required and populates them with the default value of 1. Users can adjust these default values to improve optimization.

8.5 Running models

8.5.1 R console

The steps for running models within the R console are:

1. Follow the steps above for setting up starting locations, alternative choice sets, and expectations.
2. Define the model and identify required and optional data with `make_model_design()`. Multiple models can be created before running estimation.
3. Run `discretefish_subroutine()`.

The `make_model_design()` function creates a list of all the data and model parameters required to run models. The function pulls the saved expected catch matrices from `create_expectations()` and generates the distance matrix from choices identified in `create_alternative_choice()`. If either the chosen or alternative location is a port then the `PortTable` must be available from the FishSET database. To define additional models, run `make_model_design()` again and set `replace = FALSE`. A table will be created in the FishSET database with each row being a distinct model. View more function details on the FishSET website.

Once all desired models have been defined and saved, run the models using the `discretefish_subroutine()` function. The `discretefish_subroutine()` function calls the model design table from the FishSET database and loops through each model specified in the `run` input. If `select.model` = TRUE, an interactive table will open when the function is completed. The function estimates the number of starting parameters required and will truncate or add on additional parameters as required. In addition, if `explorestarts = TRUE`, the function will try to identify better starting parameter values (smallest log likelihood) by exploring the starting parameter space. Users define the number of permutations to run (`space`) and how far to deviate from the parameter value (`dev`). At each permutation, each parameter estimate is randomly generated assuming a normal distribution with the starting parameter estimate as the mean and `dev` is the deviance. If `breakearly = TRUE`, the function will continue until a set of initial parameters values returns a value (not INF). If `breakearly=TRUE`, the function will run `space` interactions and the sets of parameter values with the lowest log likelihood value will be returned. Output from the explore starting parameters function will be passed to the optimization routine.

In addition, users have the option to normalize the data by rescaling. The default is to rescale vectors by the mean. To define a different function for rescaling, use `scaler.func`. Set `use.scalers = FALSE` if you do not want to normalize the data.

For conditional logit models (*logit_c*), more than one model may be run, depending on what was defined for the `expectcatchmodels` argument in the `make_model_design()`. If users select `expectcatchmodels = 'individual'` the model will be run four times using the specified starting parameter values and optimization options but with the expected catch matrix based on a) the selected temporal arguments, b) the short-term catch, c) medium-term catch, and d) long-term catch. Output will be returned for all four model runs.

Tracing information is returned to the R console at the level and frequency specified in the `optimOpt` argument. Output, including error messages and model comparison metrics, are saved to the FishSET database.

8.5.2 FishSET GUI

Location choices models can also be estimated from the FishSET GUI using the following procedure:

1. Follow the steps above for setting up starting locations, alternative choice sets, and expectations.
2. Define the model and identify required and optional data.
    i. Select the `Models` tab.
    ii. Select a `Likelihood function` and an `Optimization method`.
    iii. Define options for `Variables to include in the model`

1. Select optional data for travel-distance and alternative-specific variables. If there is no optional data, select "none" for both options.
2. Select the variable containing catch data.
3. The model cannot be saved if any of the three variable dropdown boxes are empty.
4. Select additional model-specific arguments.
5. For conditional logit model, select which expected catch/revenue matrices to include
    a. All matrices (includes all matrices in a single model)
    b. Run each matrix in a separate model
    c. Select a subset of matrices (select one more more matrices to include in a single model)

iv. Define `Model parameters`
1. The number of required initial parameter values will be calculated based on the likelihood function and data included (default value for each is 1). Starting parameter values should be changed, especially if there are concerns with convergence or speed. See likelihood function details for more information on initial parameter values.

v. Click the `Save model and add new model button`. The model must be saved even if not creating additional models.
1. Saved models can be deleted by checking the box next to the model that should be deleted. Note that the empty row does not have to be deleted.

3. Run models.
i. Click the `Run Models` button. This button will appear once at least one model has been saved.
ii. Tracing information will be reported to the R console.
iii. Note that each model can take several minutes to a half hour or more to run. Once models are done the output will be available.
4. View output in the `Compare Models` subtab.

8.6 Model optimization and fit

Optimization methods

FishSET uses the `optim()` function from the **`stats`** package for maximum likelihood estimation. Five optimization methods are provided in `optim()`, BFGS, CG, L-BFGS-B, SANN, and Brent. The function documentation provides a description of each method. By default, FishSET uses 'BFGS'. If there are model convergence issues during optimization, see Chapter 11 Tips for help troubleshooting.

Hessian matrix, standard errors, and t-statistics

The Hessian matrix is the second-order partial derivatives of the likelihood function evaluated at the maximum likelihood estimate. The inverse of the Hessian is an estimate of the variance-covariance matrix of the parameters. FishSET uses `optim()` from the **`stats`** package to calculate the Hessian matrix. The standard errors are the square root of the values on the diagonal of the inverse Hessian

matrix. The t-statistics are calculated by dividing parameter estimates by standard errors. The inverse Hessian is reported in model output.

Two error messages are associated with the Hessian; both of which indicate severe problems with the model specification. The first error message indicates that the Hessian matrix is not invertible (singular) and the second error message indicates that NaNs are produced, due to a negative value on the diagonal. A warning message will appear in the console to check ldglobalcheck. The ldglobalcheck table in the FishSETFolder contains the initial estimated global log likelihood value, initial parameter values, and log likelihood for alternative choices. To access this table use the `globalcheck_view(x)` function where `x` is the table name. *ldglobalcheck* table names contain the project name and date the model was run, for example, `pcodldglobalcheck20190610`.

If the Hessian matrix is singular, either the model must be respecified or more data is needed. The data could be sparse over some time periods or zones and the model therefore should be redesigned to account for this. A singular Hessian can also be caused by multicollinearity and users should check for correlation among variables. One common cause is accidentally including a full set of categorical variables instead of leaving one out (for example, including a dummy variable for North and South instead of just a single dummy variable for North). It may be that predictor variables are on different scales and you need to simply change the scaling of a predictor. Finally, check the covariance estimates. A lack of variance may indicate that the model is overparameterized or a random effect variable should be removed. For suggestions on improving model fit, see Chapter 11 Tips.

Resources for model fit metrics

AIC - Akaike, H. 1974. A new look at the statistical model identification. IEEE Transactions on Automatic Control 19 (6): 716–723.

AICc - Hurvich, CM and CL. Tsai. 1989. Regression and time series model selection in small samples. Biometrika 76: 297–307.

BIC - Schwarz, GE. 1978. Estimating the dimension of a model. Annals of Statistics 6 (2): 461– 464.

8.7 Model output

Model outputs are saved as a list to a table in the FishSET database named "ModelOut" (the table name is preceded by the project name). Each model run is added to the output list. For each model there are three main outputs that get saved: (1) a dataframe that contains estimated coefficients, standard error of estimates, and t-values; (2) optimization output, which includes the convergence value and Hessian matrix; and (3) model comparison metrics including AIC, AIC_c, BIC, Pseudo R^2. The dataframe that contains estimated coefficients is also saved as a .csv file to the "output" folder in the "FishSETFolder" directory. The name of the .csv file includes the date (YYYY-MM-DD) for which the model was run on, and multiple runs in a single day will overwrite the previous file (this file can be renamed manually to avoid overwriting).

8.7.1 R console

Purpose	Function	Table Name
Model Error	globalcheck_view(project)	*project*ldglobalcheck*YYMD*
Model Output	model_out_view(project))	*project*ModelOut

Model Estimates	model_params(project)	*project*ModelOut
Model Comparison Metrics	model_fit(project)	*project*ModelFit

Table 8.3 List of functions to pull model output tables stored in the FishSET database.

8.7.2 FishSET GUI

Model outputs are displaced in the `Compare Models` subtab of the `Model` tab. These tables show estimated coefficients, metrics of model fit, and error messages from running the optimization procedure.

8.8 Saving models

It may be desirable to identify the "best" or preferred models for future reference. Model selection can be done through the `discretefish_subroutine()` function (set `select.model` = TRUE), or using the `select_model()` function. Both methods produce a *modelChosen* table that is saved to the FishSET database. This table is not used in any functions.

8.8.1 R console

In the R console, an interactive data table can be opened by setting `select.model` in `discretefish_subroutine()` to TRUE or with the `select_model()` function. The interactive table contains model names and measures of model fit. Users can delete models from the table and select the preferred model by checking the `selected` box. The table is saved to the FishSET database with two new columns added, a TRUE/FALSE selected column and the date it was selected if `overwrite_table` is TRUE. The table is saved with the phrase '*modelChosen*' in the FishSET database.

8.8.2 FishSET GUI

Identifying and recording the "best" or preferred model is done by checking the *Selected* box next to the desired model in the *Measures of Fit* table in the `Model Comparison` subtab. Click the `Save Table` button to record this selection. Use the `Delete Row` button to remove a model.

8.9 Model performance and prediction

Once model parameters have been estimated and measures of model fit calculated, users can further evaluate model performance using k-fold cross validation and simulate predicted fishing probabilities for out-of-sample data.

8.9.1 Model performance

In addition to the measures of fit included in the model output, K-fold cross validation is a widely used method for evaluating the predictive performance of models. The general process of conducting k-fold cross validation is:

1. Split the primary dataset into *k*-folds, where *k* represents the number of independent groups (each observation appears in only one group).
2. Use *k* - 1 folds for training the model and estimating parameter values, while the one fold left out of the training group (i.e., testing group) is used for predictive performance evaluation.
3. Step 2 is repeated *k* times (iterations) such that each fold is used as the testing group once and used as part of the training data *k* - 1 times.
4. Then the performance metric for each iteration can be compared and an overall model performance metric can be calculated.

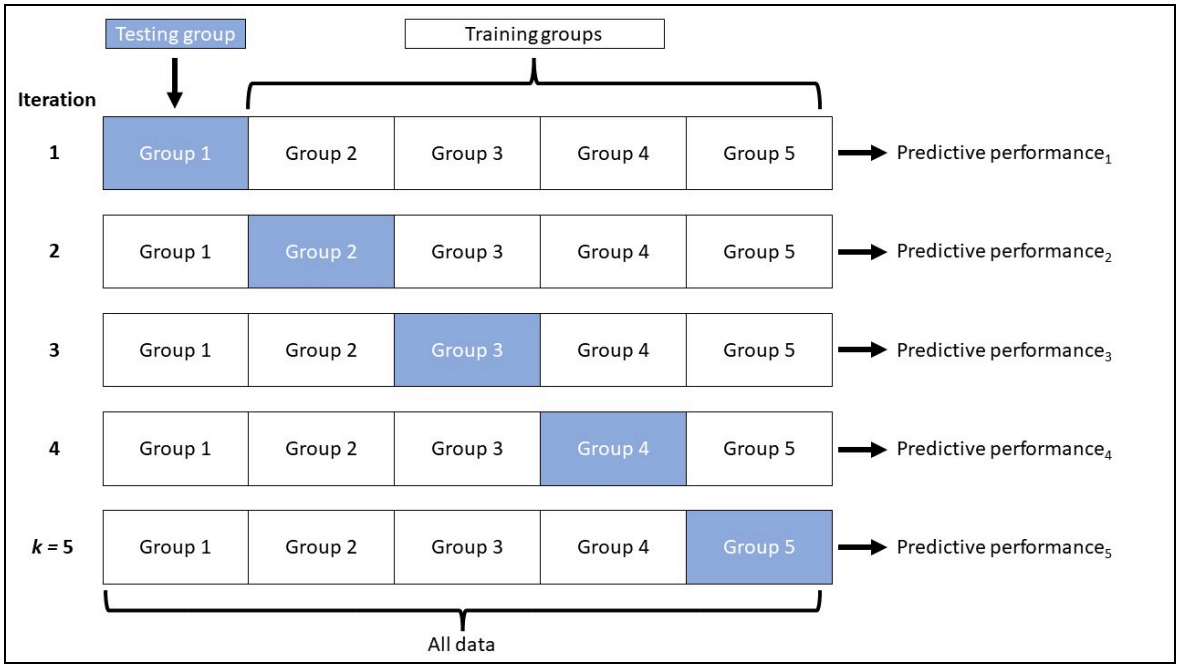


Figure 8.2 Diagram of k-fold cross validation procedure.

The `cross_validation()` function in the FishSET package allows users to conduct k-fold cross validations. There are three methods for splitting the data into training and testing groups: (1) randomly split observations into groups by setting `groups = "Observations"` and `k =` the number of groups (see note in the following paragraph on recommended values of `k`), (2) split data into groups by years by setting `groups = "Years"` and `time_var =` the name of the variable containing dates in the primary dataset, and (3) split data by a categorical variable by setting `groups =` the name of the categorical variable in the primary dataset.

For k-fold cross validations with randomly splitting observations, `k = 5` or `10` are well-established standards in the literature . Values of `k > 10` (especially for large datasets) are computationally expensive and can result in long runtimes. Cross validation that splits by years or other categorical variables may result in large differences in the number of observations across groups.

Model outputs (parameter estimates, standard errors, and t-values) and fits (AIC, AIC_c, BIC, and Pseudo R^2) for each iteration are compiled into a single .csv file in the project "output" folder and labeled as "[project]_CrossValidation.csv". The `cross_validation()` function returns a dataframe containing the percent absolute prediction error for the test group in each iteration. The percent absolute prediction error is the predictive performance metric and calculated as:

$$Percent\ absolute\ prediction\ error\ = \ 100\ * \sum_{i=1}^{J} abs\left(s_i^S - s_i^M\right),$$

and:

$$s_i^S = \frac{\sum_n^N T_{in}}{\sum_j^J \sum_n^N T_{jn}}; s_i^M = \frac{\sum_n^N E(T_{in})}{\sum_j^J \sum_n^N E(T_{jn})},$$

where J represents the number of alternatives, N is the number of observations, and s_i^S and s_i^M are the observed and predicted proportion of trips to alternative i, respectively (Klaiber and von Haefen 2019). The percent absolute prediction error represents the ability of the model to generate aggregate trip predictions that match observed data in the testing group. Note that the term "trip" is used here as an example and simply represents the choice occasion or unique observations/rows in the primary dataset.

8.9.2 Model prediction

Out-of-sample prediction (also referred to as holdout prediction) is similar to k-fold cross validation in that models are fit to training/in-sample data and predictions are estimated for testing/out-of-sample data to examine the robustness of the models prediction performance. Essentially, out-of-sample prediction is a single iteration of k-fold cross validation, but the data can be out-of-sample spatially and/or temporally. The general process of conducting out-of-sample predictions in FishSET is:

1. Load the out-of-sample dataset.
2. Filter the out-of-sample dataset.
    a. If spatially out-of-sample, then select out-of-sample locations.
3. Generate out-of-sample model designs.
4. Estimate out-of-sample predicted fishing probabilities and prediction performance.

FishSET requires out-of-sample data to be uploaded using the `load_outsample()` function *and out-of-sample data must be uploaded in the same format as the in-sample (primary) data*. Note that if data are only out-of-sample spatially and the out-of-sample zones are included in the primary dataset, then users can simply upload the main datafile again using `load_outsample()`. The functionality of `load_outsample()` is similar to `load_maindata()`.

After out-of-sample data are uploaded to the FishSET database, the out-of-sample data must be filtered using the `filter_outsample()` function. If the data are out-of-sample spatially (including spatially and temporally) then set `spatial_outsample = TRUE`, `spat =` spatial data table (same spatial file used for primary data), and `zone.spat =` the column containing zone identification in the spatial data table. When `spatial_outsample = TRUE` an interactive map will appear, which allows the user to select out-of-sample zones for prediction. If data are not out-of-sample spatially, then data are filtered for zones that were included in the selected model. *Note that models with zone-specific coefficients (e.g., zonal logit) cannot be used for predicting fishing probabilities of spatially out-of-sample data.* On completion of `filter_outsample()` a filtered dataset will be saved as a .rds file in the project "output" folder labeled "[project]filtered_outsample.rds" and existing filtered out-of-sample data files will be overwritten.

Once out-of-sample data are filtered appropriately using `filter_outsample()`, users can create model designs (refer to section 8.5 for a description of FishSET model designs) using the `model_design_outsample()` function. Set `mod.name =` the model design used for the in-sample data. The `model_design_outsample()` function will automatically pull information from the

in-sample model design to generate a new model design for the out-of-sample data, including new matrices for alternative choices and expected catch/revenue. Optional: enter a new, unique name for the out-of-sample model design using the `outsample.mod.name` input argument. If a name is not specified, the `model_design_outsample()` function will assign a default value of "[in-sample model name]_outsample". Existing model design names can be viewed in the R console by running `model_names("[project name]")`.

The `predict_outsample()` function runs model predictions and calculates model prediction performance. A list containing three elements is returned by `predict_outsample()`: (1) the mean predicted probabilities of fishing for each zone, (2) the percent absolute prediction error (equation described in section 8.8.1), and (3) predicted fishing probabilities of each zone for each observation.

9 Policy evaluation

9.1 Overview

After models are developed and parameters are estimated, FishSET can be used to analyze the effects of hypothetical or actual changes in the model inputs. Users can explore fishing effort distribution based on observational data and compare this with the estimated redistribution of fishing effort following policy changes. In addition, users can evaluate welfare loss/gain from changes in policy or changes in other factors that influence fisher location choice.

Examples of policy evaluations include the creation of area closures, the opening of new areas, changes in prices, changes in catch rates, or changes in total allowable catches. Any factor that is included in the original model can be changed and evaluated in a "policy evaluation" framework. This section provides information on evaluating different types of policy scenarios in FishSET and details on welfare analysis. In addition, at the end of the section is a table of functions related to policy evaluation.

9.2 Run policy scenarios

9.2.1 Spatial policy scenarios

To evaluate area closures and changes to zone-specific total allowable catch (TAC) in FishSET, a list of zones to be closed or a list of zones with alternative TAC is required. The `zone_closure()` function in the FishSET package allows users to easily select zones for closure or specify TAC in different zones using a GUI and save this list to the FishSET database. Once a list of zones has been saved to the FishSET database, the `run_policy()` function can be used to estimate redistributed fishing effort and welfare loss/gain from the policy scenario. Go to the FishSET website for more details on these functions. Outputs from the `run_policy()` function are automatically saved in the project "output" folder. The `pred_prob_outputs()` function can be used to display a table of redistributed fishing effort (setting `output_option = "table"`) or bar graphs of redistributed fishing effort across zones (`output_option = "model_fig"` or `"policy_fig"`). In addition, a map of redistributed fishing effort can be generated using the `predict_map()` function. Note that the welfare analysis (`welfare_predict()` function) is executed within the `run_policy()` function and outputs are saved in the project "output" folder. In the FishSET GUI, run policy scenarios and view outputs in the Policy tab.

9.3 Welfare analysis

9.3.1 Logit model welfare estimation

Welfare loss/gain for logit models (e.g., conditional logit, zonal logit) is simulated as the difference in expected surplus, which is the expected utility in dollars (Train 2002, Chapter 3):

$$W = \frac{1}{\theta}\left[ln\left(\Sigma_a^A exp\left[V_a\right]\right) - ln\left(\Sigma_b^B exp\left[V_b\right]\right)\right],$$

where θ represents the marginal utility of income, B and A refer to before and after policy change, and V is the representative utility (denominator of the conditional and zonal logit likelihoods, section 8.3.1). Estimating the marginal utility of income θ requires a cost (or revenue) variable in representative utility, and the negative of its coefficient is θ (or simply the value of the coefficient of the revenue variable). Dividing utility by θ converts the output into dollars (Train 2002, Chapter 3), which makes it possible to estimate welfare loss/gain from policy changes.

To estimate welfare loss/gain of logit models in FishSET, use the input options of the `run_policy()` function to specify which coefficient to use as the marginal utility of income θ (`marg_util_income`), and whether that coefficient relates to a cost (`income_cost = TRUE`) or revenue variable (`income_cost = FALSE`).

9.3.2 EPM welfare estimation

The estimated welfare loss/gain for EPMs is simulated using the structure of equation 10 in Haynie and Layton (2010):

$$W = \sigma_\varepsilon \times \left[ln\left(\Sigma_a^A exp\left(\frac{P_i \times E\left(Y\left(G_{im}\right)\right) - C\left(T_{in}D_{ia}\right)}{\sigma_\varepsilon}\right)\right) - ln\left(\Sigma_b^B exp\left(\frac{P_i \times E\left(Y\left(G_{im}\right)\right) - C\left(T_{in}D_{ib}\right)}{\sigma_\varepsilon}\right)\right)\right],$$

where, σ_ε is the common scale parameter (see sections 8.3.2 - 8.3.4 to see how this is used in the EPM models), P_i is the price for observation i, $E(Y(G_{im}))$ is the expected mean catch given variables G_{im}, $C(T_{in}D_{ia})$ is the cost based on the travel-distance variables (T_{in}) and distance matrix (D_{ia}), and A is the "after" policy change and B is "before" policy change. Refer back to sections 8.3.2 - 8.3.4 for more information on calculations of expected mean catch and cost for each EPM likelihood. For more information on the welfare equation, see Haynie and Layton (2010).

The welfare analysis is automatically executed within the `run_policy()` function, and outputs are saved to the project "output" folder. For EPMs, price is included in the model and users do not have to provide the marginal utility of income variable as in the logit model welfare estimation.

9.4 Data

The primary input required for policy evaluations in FishSET is the model outputs (i.e., estimated parameters) from the selected model(s). Other data requirements may vary depending on the type of policy scenarios being evaluated or models used. Additional data requirements for different policy scenarios and models are outlined below.

9.4.1 Spatial policy scenarios

In addition to model outputs, running spatial policy scenarios requires a list of zones that have been closed or a list of zones with alternative TAC per zone. This list is generated using the `zone_closure()` function.

9.5 Functions

Purpose	Function
Define closure scenarios	zone_closure()
Proposed area closures metrics	policy_metrics()
Run welfare analysis	run_policy()
Make tables and figures of policy output	pred_prob_outputs()
Generate map of redistributed fishing probabilities	predict_map()
Estimate welfare loss/gain (this is executed within run_policy() function)	welfare_predict()

Table 9.1 List of functions to run policy scenarios in FishSET.

10 Generating Reports

10.1 Report Template

FishSET includes a report template that is written in RMarkdown, which allows outputs to be easily incorporated into a report format. The report template is called *report_tempate.Rmd* and is available in the *doc* subfolder for each project directory within the "FishSETFolder" directory.

The markdown file is divided into sections, each with a series of guiding questions and suggestions for filling in the template. For example, explain how important aspects of your analysis were conducted, how was the data prepared, how were models defined, and which plots and tables to include.

The template contains guidance text that provides background to help users new to RMarkdown and examples of script. Guidance text that should be deleted before generating the report is located between the following two lines.

```
<!-- Delete text below before generating report -->
<!-- Delete text above before generating report -->
```

The report template file contains more detailed instructions for filling in each section of the template.

10.2 Report functions

There are three main types of functions that can be used to add FishSET outputs and summaries to the report: outputs functions, model summary functions, and project summary functions. While working through the template, these functions will help users add relevant outputs (figures and tables) to each section. A brief description of each type of function is described below, and more information is provided in the template markdown file.

10.2.1 Output functions

Output functions retrieve plots, tables, and notes created using FishSET and begin with the prefix *pull*. These functions retrieve files from the output folder located within the FishSETFolder > `project_name`

directory.Add *pull* functions inside a code chunk to display an output in the report document. Refer to the report template for examples on using these functions and see the FishSET website for documentation.

10.2.2 Model summary functions

The model summary functions pull and display model output saved to the FishSET database. Given a project name, `model_out_summary()`, `model_error_summary()`, and `model_fit_summary()` display model output, error, and fit tables, respectively. Go to the FishSET website to view function documentation.

10.2.3 Project summary functions

Project summary functions display the various database tables, files, and function calls that were created in the project. These are output that you may want to share or reference but are more detailed than traditionally required for a report. The output for these functions appear in the *Appendix* section of the report. The project summary functions are detailed below, however, no steps beyond defining the `project` and `date` fields in the YAML header are required.

Appendix A: Project Notes	`pander(pandoc.list(pull_notes(params$project))`
Appendix B: Log Summary	
Data Loading	`pander(function_summary(date = params$date, type = "dat_load",show = 'last')`
Data Quality Evaluation	`pander(function_summary(date = params$date, type = "dat_quality", show = "last")`
Data Exploration	`pander(function_summary(date = params$date, type = "dat_exploration", show = "last")`
Fleet Assignment and Summary	`pander(function_summary(date = params$date, type = "fleet", show ="last")`
Zonal Definition	`pander(function_summary(date = params$date, type = "zonal_def", show = "last"))`
Models	`pander(function_summary(date = params$date, type = "model",show = "last"))`
Appendix C: Project database tables	`project_tables(params$project)`
Appendix D: Project file output (package output folder)	`project_files(params$project`

10.3 Generating reports

The FishSET report template generates PDFs by default (see "Report options and setup" section above for changing output type). However, R Markdown is capable of creating other types of documents, including Microsoft Word. Generating PDF documents requires a LaTeX engine which can be installed by running the code below:

```
```{r}
 install.packages("tinytex")
 tinytex::install_tinytex()
```
```

10.3.1 Additional markdown resources

Generating MS word documents	https://bookdown.org/yihui/rmarkdown/word-document.html
Generating PDF documents	https://bookdown.org/yihui/rmarkdown/pdf-document.html
R Markdown cheatsheet	https://rstudio.com/resources/cheatsheets/

10.4 Other reproducible workflow options

FishSET has several features to aid in transparency and developing a reproducible workflow. These include:

1) Saving all data used in analyses in the FishSET database, including saving a raw, unmodified version.
2) Recording all function calls in log files.
3) Saving all output (figures, tables, and messages).
4) Including functions to pull output into the report template.

In addition to these features, researchers may wish to work in Jupyter Notebooks or RMarkdown Notebooks. Working in a notebook file allows users to run functions while saving function calls, notes, and observations in a single file that can be shared as a working analytical file. All functions run in a notebook file will be logged and output saved in the *Output* folder, allowing for the script to be easily rerun and output to be incorporated into the report template. The greatest advantage to working in a Notebook file is that it makes it easier to work collaboratively on analyses, especially with collaborators not familiar with R.

Jupyter Notebooks can be used in both a standard R terminal and an RStudio development environment. RMarkdown Notebooks can only be used in an RStudio development environment.

11 Tips

11.1 Handling large datasets

Big data is a special challenge in FishSET. Functions take longer to run, especially in the FishSET GUI, than smaller datasets. Trends and patterns may be harder to detect with cluttered plots, making the data less efficient for data exploration. There is also the risk of running out of memory space, especially when running models with numerous fishery or regulatory zones.

FishSET functions have been written to effectively handle larger datasets as much as possible. These include vectoring code and saving intermediate data in the FishSET database rather than local memory. However, it is not possible to accommodate all potential issues with handling big data, especially with data exploration and visualization.

If data size is an issue we recommend:

> Randomly select a subset of the data for data exploration and model design. Run the best model with the full dataset.
>
> Remove ancillary data from the main dataset. This may include duplicate variables such a homeport variable coded as homeport name and homeport code. Auxiliary data that is not needed, such as vessel characteristics, can be split from the primary dataset and saved. Data from areas outside the area of concern could also be removed. See Chapter 4 identifying messy data for more ways to simplify the data.
>
> If possible, split the data into logical subparts, such as by species. Run individual models for each subset of the data.

11.2 Missing Values

Missing values are not allowed in model data. Missing values must be removed or extrapolated. Generally, variables with a high frequency of missing data should be removed. If only a limited number of rows of the data frame contain missing data points and data are missing systematically, then the missing data can be removed or extrapolated. Missing data from the location choice variable cannot be extrapolated. See `na_filter()` in Chapter 4 Missing values for more details on how to assess occurrence of missing values and replacing missing values with the mean or median of the variable.

11.3. Data sparsity

Data sparsity is an interesting issue in modeling data. On the one hand, we want to know why a rare event occurred or to know more about a sparse variable. On the other hand, it is hard to make inferences about the rare event without sufficient data. For instance, we don't know if fishing is not observed in an area because of preference to fish in recently fished areas or because there is something undesirable about the unchosen area.

Users should look at the distribution of observations across zones and decide if a minimum should be applied and what that value should be. In the R console, `sparsetable()` and `sparseplot()` function will calculate sparsity by time and space. See the FishSET website for function documentation.

Sparsity is also a consideration when designing how estimating expected catch or revenue is calculated. There is frequently limited data over some time periods or some areas. For example, using near-term catch (past 2-3 days) to estimate catch will result in missing values if catch data are sparse so it may be preferable to calculate averages over the past seven days or more. In the FishSET GUI, the sparsity table and plot is displayed in the Expected Catch/Revenue tab. The output will appear only after the `Temporal variable for averaging` is identified.

11.4 Model convergence issues

Below is a set of suggested approaches for addressing poor model convergence.

- Were any convergence errors returned by the optimization function?

```
model_error_summary(project, output='print')
```

- Check sparsity in the expected catch matrix.
- Evaluate the Hessian matrix.

    - The Hessian should be negative definite.
- Check the ratio of the largest to smallest eigenvalues of the Hessian (the condition number).
    - If the condition number of the matrix exceeds 1/ε, where ε is machine precision, the solution is likely to be unreliable.
        - 1/Ɛ is approximately $4.5e^{15}$ on a PC.
- Inspect the successive log-likelihood values of the iterations
    - The sequence should show movement toward a single value. If the difference between log-likelihood values does not decrease with each iteration then the model has probably not converged on a minimum.
- Profile the likelihood to assess the adequacy of the quadratic approximation.
    - This can be time intensive.
    - Rerun the model with different starting parameters. If the parameter estimates are radically different then the optimization surface likely has multiple local maxima.
- Check the estimated parameters are within the parameter bounds.

11.5 Troubleshooting fitting models

There are a few procedures that can be taken to improve model fitting.

- Scale the independent variables so that all variables should be on similar scales. For example, try using thousands of pounds instead of pounds as the units of catch. Another method is to standarize: subtract the mean and divide by the standard deviation.

```
x-mean(X)/sd(X)
```

- Parameter starting values should be within the bounds of the variable data. If data are standardized, then the starting parameter should be within the bounds of the standardized data, not the unstandardized data.
- Use informed parameter starting values. Highly accurate starting values are generally not needed, but really wild or lazy choices can lead to slow or poor model convergence.
    - Parameter estimates from a linear approximation as starting values
    - Build and run simple models to get starting parameter values to plug into more complex models.
- Check for multicollinearity. A number of issues can arise if variables are correlated, including more convergence, singular Hessian matrix, and unreliable results. Users should remove the correlated variables with the weakest effect on the explanatory variable.
- Try a different optimization method.

11.6 Logged function calls

Functions are saved in log files in the *src* folder in project directory of the FishSETFolder directory. Entire log files can be removed manually.

11.7 Tables and plots

Tables and plots are saved in the *Output* folder in the project directory of the FishSETFolder directory. These files must be removed manually.

11.8 Will FishSET work on a mac?

Yes, FishSET will work on a mac. However, mac users often experience an rJava issue with RStudio. rJava is a simple R-to-Java interface that allows for creating objects, calling methods, and accessing fields of Java objects from R.

If you see an error message similar to the one below when trying to load rJava, please read on for a detailed description of one possible solution.

```
> library(rJava)



.onLoad failed in loadNamespace() for 'rJava', details:

  call: dyn.load(file, DLLpath = DLLpath, ...)

  error: unable to load shared object
'/Library/Frameworks/R.framework/Versions/3.6/Resources/library/rJava/libs/rJava.so':


dlopen(/Library/Frameworks/R.framework/Versions/3.6/Resources/library/rJava/libs/rJava
.so, 6): Library not loaded:
/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/lib/server/libjvm.dylib

  Referenced from:
/Library/Frameworks/R.framework/Versions/3.6/Resources/library/rJava/libs/rJava.so

  Reason: image not found
```

If rJava fails to load, then it's unlikely that you will be able to run FishSET. There is no single solution to this problem and all solutions involve running commands in the Terminal.

The terminal can be accessed in RStudio through the *Terminal* tab next to the *Console* tab or by going to Applications --> Utilities --> Terminal.

A Java Development Kit (JDK) must be installed for rJava to work properly. To check if a JDK is installed, run `o java -version` in the terminal. If a JDK is installed, the terminal should output the java version and date of installation:

```
Smith-MBP:FishSET_binary JohnSmith$ java -version

java version "11.0.6" 2020-01-14 LTS

Java(TM) SE Runtime Environment 18.9 (build 11.0.6+8-LTS)

Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.6+8-LTS, mixed mode)
```

If a JDK is not installed, it will need to be downloaded from https://www.oracle.com/java/technologies/javase-downloads.html. Install version 8 or 11+ but avoid

version 9. After JDK has been installed, run `sudo R CMD javareconf` in the terminal. Finally, restart RStudio and run `library(rJava)` in the R console.

If the above solution does not work, try following the steps below, as described on a GitHub discussion page.

1. Check if you have at least one version of JDK installed.

2. Check if you have set the JAVA_HOME variable.
   In the terminal, run echo $JAVA_HOME

   If there is no path:
   run open .bash_profile
   edit text in export JAVA_HOME=`/usr/libexec/java_home`

   If there are multiple versions of JDK:
   export JAVA_10_HOME=$(/usr/libexec/java_home -v10)
   export JAVA_11_HOME=$(/usr/libexec/java_home -v11)
   alias jdk10='export JAVA_HOME=$JAVA_10_HOME'
   alias jdk11='export JAVA_HOME=$JAVA_11_HOME'
   export JAVA_HOME=`/usr/libexec/java_home`

   Save it and run source .bash_profile to apply the change.

3. Restart the terminal and run echo $JAVA_HOME and check the path is listed.

4. In the terminal, use sudo R CMD javareconf to reset R's java environment configuration.

5. In the R console run options("java.home"),

   If the answer is NULL, run:
   options("java.home"="/Library/Java/JavaVirtualMachines/jdk- 11.0.4.jdk/Contents/Home")
   Paste your own java_home path in the quote.

6. If, after following the steps above, check the version part of the error message.
   /Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/lib/server/libjvm.dylib
   For instance, the version part in the error message above, 11.0.1, does not match the JDK version 11.0.4 specified in step 5. It's possible the script from Rstudio, which is rJava.so, did not use the java path given. It detects your java version and loads the fixed version of JDK lib. And for JDK version 11, this is jdk-11.0.1.

   A little trick:
   Before you run library(rJava) in Rstudio's Console, temporarily change your jdk-version folder's name

to 11.0.1 (if you are using jdk11). Other versions may also be fixed like this. Then it may work! Don't forget to change the folder's name back!

11.9 How do I edit data?

Data may need to be edited for a number of reasons such as data imputation errors or inconsistency in reporting units.

Data can be edited outside R, in the user's preferred program, such as Excel, and then imported into FishSET. If the data was imported into FishSET before editing then the data table should be replaced or removed from the FishSET database. To replace the table, use the argument `over_write = TRUE` in `load_maindata()`. To remove the table before importing, use `table_remove('projectMainDataTable', 'project')` where *project* is the project name.

Data loaded into the R working environment can be edited directly in RStudio Figure 17.1. Note that this method is slow. First load the data into the working environment. If the data table exists in the FishSET database use

```
projectMainDataTable <- load_data('project')
```

where *project* is the name of your project. If the data is not in the FishSET database, use `read_dat()`. Available datasets will appear under 'Data' in the top right hand corner of RStudio. To edit, type

```
projectMainDataTable <- edit(projectMainDataTable).
```

A Data Editor window will open. After making changes, close the window. The changes will be applied to the data table.

Finally, the data table can be edited directly in the FishSET GUI (Figure 17.2). Open the `Data Exploration` tab. If the data table is not displayed, select *`Table`* under `View data table or plots`. Double click on any cell to edit. Click anywhere outside the cell when done. Press the white `Save data to FishSET database` button to save the revised data table.

11.10 Running R code in the GUI

R code can be run directly from the GUI by selecting "Other actions" in the left panel on most tabs. In the "Enter an R expression" input box, write R code as you would in a script or in the console, then click "Run". The primary data is loaded into the session as "values$dataset", so if you want to access a variable named "Var1" from the primary table then you would write "values$dataset$Var1".

11.11 Data did not save to FishSET database

There are four primary reasons that data would fail to be saved to the FishSET database.

1. **The data is a spatial dataset**. Spatial datasets (i.e., shape files) cannot be saved to the SQLite FishSET database.
2. **Data checks fail**. Several checks are made when data is loaded into FishSET that must be passed for data to be saved to the FishSET database. Warning messages in the R console should indicate which check did not pass. These data quality issues must be addressed before the data can be saved to the FishSET database.

3. **Table already exists** in the FishSET database. To replace an existing table in the database, use `over_write = TRUE` when loading the data. To save the table without overwriting, define a new project name.
4. **Data import error**. Data may not save to the database because it has not been properly imported into the R environment. The FishSET function to read in data (`read_dat()`) is a wrapper that calls the most flexible function to import the data based on the data file extension. An error will occur if further arguments are needed to correctly import the data. For example, if metadata is included above the data in an excel spreadsheet, the argument `skip_lines` needs to be included in the `read_dat()` function call for the data to be imported properly. Error messages in the R console should indicate why the data could not be imported. Users can address issues directly in the data or add further arguments to the `read_dat()` function call. Details on further arguments that can be called are in the function documentation. The `read_dat()` function documentation provides links to the relevant documentation for functions called by `read_dat()`.

11.12 Map is in the wrong location or data locations not assigned to grid

Check that the coordinate variables have the correct signs and format (decimal degrees). Next, check that the correct spatial file was loaded and that there were no warning or error messages when importing the data.

11.13 Getting Help

Use resources on the web including Stack Overflow and R-Help. You can ask questions on these sites and other R users will try to answer them. The easiest way to search questions online is to type 'R-help' followed by the question in your browser search bar, which will find questions on the topic in both Stack Overflow and R-Help. ChatGPT can also be a good resource for R help.

12 References

R

RStudio Team. 2020. RStudio: Integrated Development for R. RStudio, PBC, Boston, MA
URL *http://www.rstudio.com/*

Modeling Papers

Abbott, JK and JE Wilen. 2011. Dissecting the tragedy: A spatial model of behavior in the commons. Journal of Environmental Economics and Management 62(3): 386-401. https://doi.org/10.1016/j.jeem.2011.07.001.

Dahl, G. 2002. Mobility and the returns to education: testing a Roy Model with multiple markets. Econometrica 70(6): 2367-2420.

Girardin, R, KG Hamon, J Pinnegar, JJ Poos, O Thebaud, A Tidd, Y Vermard and P Marchal. 2017. Thirty years of fleet dynamics modelling using discrete-choice models: What have we learned? Fish and Fisheries 18(4): 638-655. https://onlinelibrary.wiley.com/doi/abs/10.1111/faf.12194

Haynie, AC, RL Hicks, and KE Schnier. 2009. Common property, information, and cooperation: Commercial fishing in the Bering Sea. Ecological Economics. 69(2): 406-413. https://doi.org/10.1016/j.ecolecon.2009.08.027

Haynie, AC and DF Layton. 2010. An expected profit model for monetizing fishing location choices. Journal of Environmental Economics and Management 59(2): 165-176. https://doi.org/10.1016/j.jeem.2009.11.001.

Hilborn, R. 2007. Managing fisheries is managing people: what has been learned? Fish and Fisheries 8: 285–296.

Hicks, RL and KE Schnier. 2008. Eco-labeling and dolphin avoidance: A dynamic model of tuna fishing in the Eastern Tropical Pacific. Journal of Environmental Economics and Management 56(2):103-116. https://doi.org/10.1016/j.jeem.2008.01.001

Peterson, G. 2000. Political ecology and ecological resilience: An integration of human and ecological dynamics. Ecological Economics 35: 323–336.

McFadden, D. 1974. The measurement of urban travel demand. Journal of Public Economics 3:303– 328.

Raphael, G, K Hamon, J Pinnegar, JJ Poos, O Thébaud, A Tidd, Y Vermard, and P Marchal. 2017. Thirty years of fleet dynamics modelling using discrete choice models: What have we learned? Fish and Fisheries 18: 638-655.

Smith, MD and JE Wilen. 2003. Economic impacts of marine reserves: the importance of spatial behavior. Journal of Environmental Economics and Management 46(2): 183-206. https://doi.org/10.1016/S0095-0696(03)00024-X.

Mac Help

GitHub page https://github.com/rstudio/rstudio/issues/2254#issuecomment-537175167

Optimization and Model selection metrics

Belisle, CJP. 1992. Convergence theorems for a class of simulated annealing algorithms on R^d. Journal of Applied Probability 29:885–895. doi: 10.2307/3214721.

Byrd, RH, P Lu, J Nocedal and C Zhu. 1995. A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing 16:1190–1208. doi: 10.1137/0916069.

Fletcher, R and CM Reeves. 1964. Function minimization by conjugate gradients. Computer Journal 7:148–154. doi: 10.1093/comjnl/7.2.149.

Nash, JC. 1990. Compact Numerical Methods for Computers. Linear Algebra and Function Minimisation. Adam Hilger.

Nelder, JA and R Mead. 1965. A simplex algorithm for function minimization. Computer Journal 7:308–313. doi: 10.1093/comjnl/7.4.308.

Nocedal, J and SJ Wright. 1999. Numerical Optimization. Springer.

Model averaging

Claeskens, G. and NL Hjort. 2008. Model Selection and Model Averaging, Cambridge: Cambridge University Press.