FinCatch Documentation

Keith Hurley et al

Invalid Date

Table of contents

Home **5**

6

I FinCatch System **7**

1 System Components **8**
1.1 FinCatch . . . 8
1.2 FinCatch Database . . . 8
1.3 FinCatchDE . . . 8
1.4 FinCatchAG . . . 8
1.5 FinCatchRA . . . 8
1.6 FinCatchAnalysis R Package . . . 8
1.7 FinCatchAccess R Package . . . 8
1.8 FinCatchWebApi . . . 8

2 Sampling Restrictions **9**

3 Data Design **10**
3.1 Processed Species . . . 10
3.2 Extrapolation of Length Subsampling . . . 10

4 Architecture **11**
4.1 Packages . . . 11
4.2 WorkFlow . . . 11
4.3 Principles . . . 12

II Analysis R Package **13**

5 Overview **14**
Overview . . . 14
5.1 Analysis Functions . . . 14
5.2 Analysis Return Objects . . . 14

6 Example Script **15**

7 Base Object **16**
7.1 Purpose . . . 16
7.1.1 Variables . . . 16
7.1.2 Methods . . . 17
7.2 Overriding Base Object . . . 17

8 Objects **18**
8.1 fc_counts Object . . . 18
8.2 fc_fishSampleMedtadata . . . 18

9 Methods **19**
9.1 fca_counts . . . 19
9.2 fca_fishSampleMetadata . . . 19

10 Helpers **20**
10.1 Dplyr Verbs . . . 20
10.1.1 AddStockCategory . . . 20
10.1.2 AddWrParameters . . . 20
10.1.3 AddAgeIntercept . . . 20
10.2 Functions . . . 21
10.2.1 weighted.se.mean . . . 21
10.2.2 fca_getAnalysisFunctions . . . 21
10.3 Labelers . . . 21
10.3.1 fc_labeler_Survey . . . 21
10.3.2 fc_labeler_fishSample . . . 22
10.3.3 fc_labeler_wqSample . . . 22
10.3.4 fc_matchCodes . . . 22
10.3.5 fc_createCodeLabel . . . 22
10.3.6 fc_createCodeLabelReversed . . . 22

11 Create Analysis **23**
11.0.1 General . . . 23
11.0.2 Steps . . . 23
11.0.3 Create Custom Analysis R6 Object . . . 25

III Data Access R Package **27**
Packages . . . 28
WorkFlow . . . 28
Principles . . . 28

IV References **30**

Home

This is documentation for the FinCatch Data System of the Nebraska Game and Parks Fishery Division. FinCatch stores and provides analysis of standard lentic fisheries population surveys. This set of documentation is an accumulation of both developmental and instructional documentation.

Part I

FinCatch System

1 System Components

The FinCatch data system is comprised of a number of separate components, including:

1.1 FinCatch

FinCatch is the central website that provides data management capabilities as well as links to other components. FinCatch is written in the asp.mvc framework of .NET 6.

1.2 FinCatch Database

The backend database for FinCatch is built in Microsoft SQL Server.

1.3 FinCatchDE

1.4 FinCatchAG

1.5 FinCatchRA

1.6 FinCatchAnalysis R Package

1.7 FinCatchAccess R Package

1.8 FinCatchWebApi

2 Sampling Restrictions

- All length subsampling should occur at the SAMPLE level.
- All weight and age subsampling may occur at the SAMPLE or SURVEY level.
- Every individual of a species in the processed species list has either been measured or counted. If "All Species" (code 0) is in the processed species list, EVERY individual encountered during sampling was either measured or counted.
- Odd species (those not found in the processed species list) may be measured or counted and entered into FinCatch. However, they will only be reported in raw data summaries and NOT included in calculations and analysis as they cannot be assumed to have been collected in their entirety (i.e. all individuals in all samples).
- If the "Use For CPUE" flag is FALSE (unchecked), then NONE of the data for any species in the sample will be used for catch rate calculations. If the flag is TRUE, then ALL species in the sample will be included for catch rate calculations.

3 Data Design

3.1 Processed Species

A list of species processed during sampling is collected at the survey level and applies to all fish samples attached to the survey. Any species included in the processed species list and NOT found in the data carries the assumption of a zero-catch. If a species is found in the data and NOT in the process species list, no assumption is made that the species was consistently processed throughout the survey.

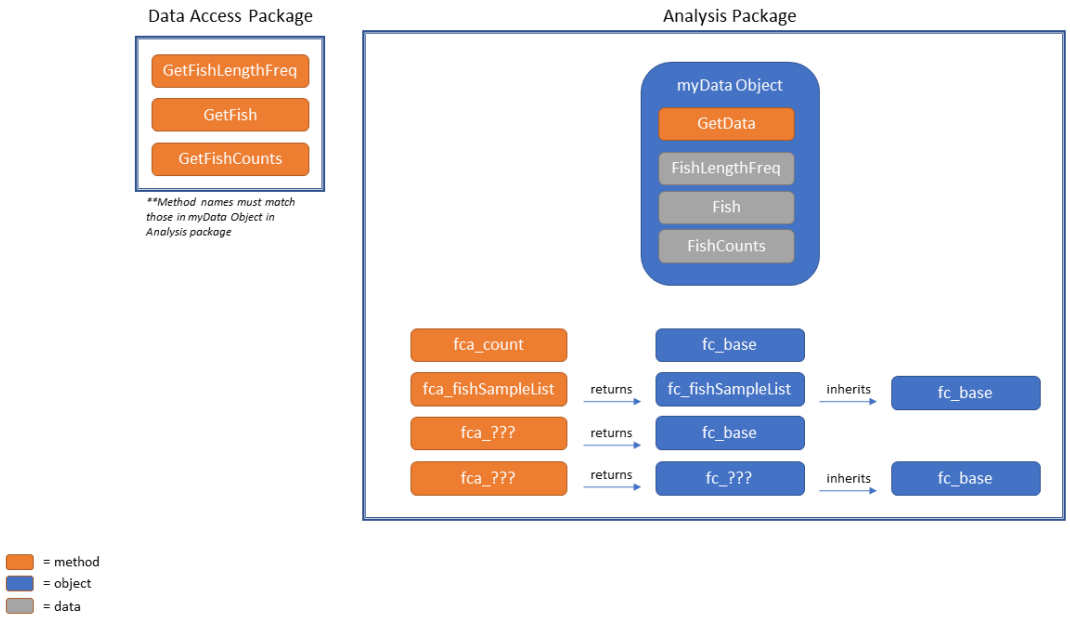
All species (code 0) can be included in the processed species list. The inclusion of this code carries the assumption that ALL fish collected during the survey (in ALL samples) were processed. During analysis, the processed species list will be expanded to include all species found in the data.

> **Note**
>
> Important caveat: expanded to include all species in the analysis and not just in the survey. This allows the following example to work: all Omaha area surveys were analyzed for red ear population structure - but redear were only found in half the reservoirs. By expanding the processed species list by analysis data rather than grouped within survey, zero's will be included for redear in all surveys in the analysis that have all species (code 0) in the processing list.

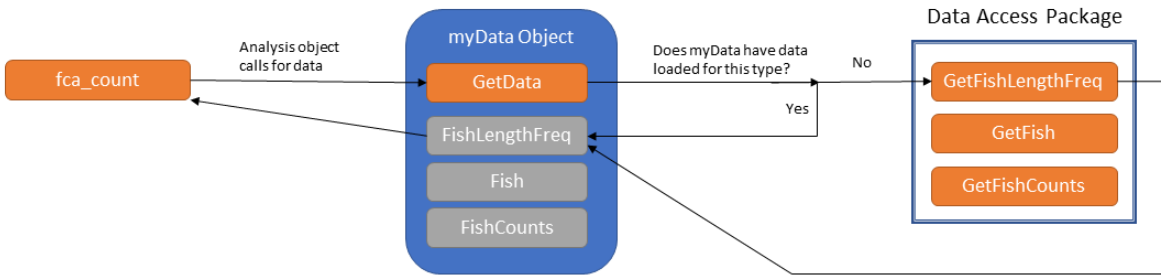
3.2 Extrapolation of Length Subsampling

4 Architecture

4.1 Packages



4.2 WorkFlow



4.3 Principles

By Using myData Object:

- Only data that's needed is loaded
- Data is cached
- Data is only retrieved once for entire analysis string

By Using Separate Data Access Package:

- Isolates and generalizes data access
- Allows analysis code to use different sources of data
- Allows different scripts to use different authentications for data access

Part II

Analysis R Package

5 Overview

Overview

The FinCatchAnalysis (FCA) R package centralizes standard analysis functions for the FinCatch system and promotes DRY and reusable analysis code practices. The FCA package is built on top of R6 classes which provides a standard programming interface for users of the R package. Analysis functions are available for each individual analysis available and results from each analysis function returns results encapsulated in an R6 class object. All public functions in the package are prefixed with "fca_". Package R6 objects are prefixed with "fco_".

5.1 Analysis Functions

Each analysis function in the FCA package provides a single call for an independent analysis and returns a function specific R6 object built on the base fco_ object that provides standard methods and data objects.

5.2 Analysis Return Objects

6 Example Script

```
library(FinCatchAccess)
library(FinCatchAnalysis)

#use filters gadget to make a filter object
myFilters<-fcacc_show_FilterSelector_gadget()

#feed the filter object with your ID values into a data object
myDataObject<-fc_data$new(myFilterObject=myFilters, myGroupSurveys=TRUE)

#run an analysis by feeding it the data object
myAnalysis<-fca_meanLength(myDataObject)

#print output tables with printTablesAuto/Html/Latex
myAnalysis$printTablesAuto()

#if you want canned plots
myAnalysis$plots
```

7 Base Object

7.1 Purpose

Provides a common interface for results regardless of what analysis is run.

7.1.1 Variables

- analysisTitle - (string) Description of the analysis stored in the object, used for headers and titles in reports and displays
- exportName - (string) Name used when generating file download names, no spaces or punctuation should be used
- descriptionText - (string) Markdown text to be displayed in outputs prior to the output tables and figures
- tableTitle- (list<string>) Title text to be used on object tables, each item in list corresponds to one table output in printTables
- groupByVars - (string) Comma-separated string of variable name to use as grouping variables within the tables
- surveys - (dataframe) Dataframe containing survey-level data
- samples - (dataframe) Dataframe containing samples
- results - (list<dataframes>) List of dataframes containing results of analysis
- plots - (list<ggplot objects>) List of ggPlot objects created from analysis
- errors - (list<string>) Error messages created during analysis
- warnings- (list<string>) Warning messages created during analysis
- groupHeaderBackgroundColor - (string) Color to be used in background of group header within table, groups are determined by the groupByVars variable, used in HTML outputs only

- groupSummaryBackgroundColor - (string) Color to be used in the summary row of each group within table, groups are determined by the groupByVars variable, used in HTML outputs only
- gtTheme - (string) Name of gt tables theme from the gtExtras package, used in HTML outputs only

7.1.2 Methods

- print - method used by R to print the results variable
- exportJson - method used to create and save Json file of results, no default implementation
- exportCsv - method used to create and save Csv file of results, no default implementation
- createTable - generic table to create both Latex and Html tables for object
- createTableLatex - creates table formatted for Latex, default implementation simply calls and returns createTable
- createTableHtml - creates table formatted for Html, default implementation simply calls and returns createTable
- printTablesLatex - called to output all tables in a LaTex format, this does not include any formatting or table specific code which is included in the createTable? functions, but instead iterates through multiple table outputs
- printTablesHtml - called to output all tables in a Html format, this does not include any formatting or table specific code which is included in the createTable? functions, but instead iterates through multiple table outputs
- printTablesAuto - tests the incoming call for a LaTex environment and calls either printTablesLatex or printTablesHtml as appropiate, used for markdown reports that can be user-generated in mulitple formats
- iterateSurvey - a generic function that crawls through two loops, one for survey groups, one for specific tables (one analysis may output mulitiple tables) and is called by the printTable functions
- printPlots - called to output ggplot outputs stored as part of analysis

7.2 Overriding Base Object

The fc_base object can be extended and customized to produce objects for specific analysis. See details in “Creating Analyis Chapter”.

8 Objects

8.1 fc_counts Object

8.2 fc_fishSampleMedtadata

9 Methods

9.1 fca_counts

- summarizes number of fish caught in samples by species and gender
- returns fc_counts object

9.2 fca_fishSampleMetadata

- returns metadata for fish samples included in analysis
- returns fc_fishSampleMetadata object

10 Helpers

There are a number of helper functions available in the FinCatch Analysis package.

10.1 Dplyr Verbs

10.1.1 AddStockCategory

This function works as a dplyr verb and adds a column containing a factor of stock-length categories from Gabelhouse's 5-cell model. The function takes the name of the column containing the species code, the column containing the fish length (in mm), and a reference to the data object for the analysis (needed to retrieve species code and stock length values). The function defaults to full group names as factor labels; use abbrieviations=TRUE to use category abbrieviations instead.

Example:

```
some_data %>%
  AddStockCategory(sppCode, fishLen, someDataObject, useAbbreviations = TRUE)
```

10.1.2 AddWrParameters

This dplyr verb has yet to be implented

10.1.3 AddAgeIntercept

This dplyr verb has yet to be implemented

10.2 Functions

10.2.1 weighted.se.mean

This function calculates standard error around a weighted mean. The function assumes a weighted mean calculated by the "weighted.mean()" function in the stats package of R and accepts the same arguements.

Example:

```
some_data=data.frame(xValue=c(3,4,6,3,2,3),
                     xWeights=c(0.2, 0.2, 0.3, 0.1, 0.1, 0.1))
some_data %>%
  mutate(myWeightedMean=weighted.mean(xValue, xWeights, na.rm=TRUE),
         myWeightedMeanSE=weighted.se.mean(xValue, xWeights, na.rm=TRUE))
```

10.2.2 fca_getAnalysisFunctions

This function returns a list and description of the analysis functions available in the FinCatch-Analysis package.

Example:

```
fca_getAnalysisFunctions()
```

10.3 Labelers

10.3.1 fc_labeler_Survey

This function creates a label for each survey to be used in outputs such as reports and plots. The arguments are a vector of surveyUid values and a reference to the analysis data object. The label returned is structured like:

> Title1 (WB=5110 | Method=21 | Year=2022 | Season=Spring)

Example:

```
some_data %>%
  mutate(surveyLabel=fc_labeler_Survey(surveyUid, aDataObject))
```

10.3.2 fc_labeler_fishSample

This function creates a label for each fish sample to be used in outputs such as reports and plots. The arguments are a vector of surveyUid values and a dataframe of fish samples in the analysis taken from the data object. The label returned is structured like:

> WB=2832 | Method=45 | 2022-03-11 | Station=627

Example:

```
some_data %>%
  mutate(sampleLabel=fc_labeler_fishSample(surveyUid, aDataObject$get_data_samplesFish))
```

10.3.3 fc_labeler_wqSample

This function creates a label for each fish sample to be used in outputs such as reports and plots. The arguments are a vector of surveyUid values and a dataframe of fish samples in the analysis taken from the data object. The label returned is structured like:

> WB=2832 | 2022-03-11 | Station=627

Example:

```
some_data %>%
  mutate(sampleLabel=fc_labeler_wqSample(surveyUid, aDataObject$get_data_samplesWq))
```

10.3.4 fc_matchCodes

www

10.3.5 fc_createCodeLabel

www

10.3.6 fc_createCodeLabelReversed

www

11 Create Analysis

11.0.1 General

Analysis function calls are prefixed with "fca_" and object names are prefixed with "fc_"

Make sure to test each analysis function for:

- Data selected by surveys only
- Data selected by samples only
- Data selected by both surveys and samples
- Filters that return NO data
- Species in processing list and not in data
- Species not in processing list and in data
- Works for both grouped by survey and ungrouped analysis, if not grouped by survey in fc_data$groupSurveys, surveyUid is set to "-1" during data download by the data object

11.0.2 Steps

1. Create new r file names fc_analysisName.R (Easiest To Copy Existing Analysis and Modify)
2. No library statements should be included in R file. Instead, they need to be included in the package DESCRIPTION file.
3. Create/Modify the roxygen comments for procedure
4. Name/Rename function. Analysis functions are prefixed with "fca_" and the same base name as the "fc_" file.
5. All function calls require a FinCatch Data Object (fc_data) to be passed to an arguement named "myData"
6. Check that fish (or Wq) samples exist in the current dataset.
7. Set grouping variables.

- To group analysis calculations during analysis, use the dplyr verbs

```
group_by(across(all_of(myGroups))
```

- add addition fields as necessary "group_by(across(all_of(myGroups), anotherFieldHere)"

8. Write analysis code

- Always include Standard Error and Sample size (if appropriate), this allow users to calculate difference confidence intervals post hoc
- When including confidence intervals, include 95% and 80%
- Make sure to account for missing data in any input

9. Label values like survey, sample, species, waterbody, etc.

- Labelers exist for samples and surveys (make sure surveyUid's are set to -1 if not grouping by survey
- Helper functions are available for coded values

10. If analysis is calculated (as opposed to raw data summations), screen out species NOT in the processing list. This is necessisary because species NOT in the list do not carry the assumption that all specimans for that species were processed.

11. Attach all results to an analysis object (either base or custom). Example:

```
#create return object at beginning of function
(this allows errors and warnings to be added throughout survey)
op <- fc_meanWeight$new()
op$analysisTitle = "Mean Weight (Weighted)"
op$exportName = "MeanWeight"
op$descriptionText = "These results display weighted mean
                      weight.  Calculations are weighted as
                      only a non-proportional (i.e. first 10 fish
                      per 10mm length group) number of fish
                      are subsampled."
op$tableTitle <- list("Weighted Mean Weight")
op$groupByVars <- "speciesCode"
op$groupSurveys <- myData$groupSurveys

#during function, add warnings and errors to return object
op$errors <- rlist::list.append(op$errors, "Error1 Message", "Error2 Message")
```

```
op$warnings <- rlist::list.append(op$warnings, "Warning1")

#at end of function, add results to return object
op$results <- list(data.frame(d))
op$plots <- myPlots
```

12. Add the function to the list of analysis functions available in the package (found in inst folder). This list is used to populate the FinCatchRA UI and to feed the fc_getAvailableAnalysis function.

11.0.3 Create Custom Analysis R6 Object

All analysis results are returned using R6 objects. This allows for consistent use and implementation of different analysis functions by parent applications and code. A base R6 object, "fc_base", provides all the basic functions and structure for FinCatch analysis objects. Custom objects can be created in the analysis files to allow customized output tables and plots and MUST inherit from the fc_base object.

Column names will often need to be altered to provided user friendly text in the outputs. In addition, sometimes valuable columns are dropped for display purposes. Both of these should be done in the analysis output object createTable functions, NOT in the analysis function itself or in the "results" property of the output object. This is to provide for consistency between analyses and is important for the download function of FinCatchRA.

Every effort is made to ensure tables produced by analysis objects work in both HTML (which allows more formatting options and used by FinCatchRA) and in LaTex (for PDF report generation). Basic table creation should happen by providing a "CreateTable" function. Any additional work needed for specific HTML or LaTex output should be included in overridden "CreateTableHtml" or "CreateTableLatex" functions….which otherwise just call and return the "CreateTable" function by default.

```
fc_counts <- R6Class("fc_counts",
                     inherit = fc_base,
                     public = list(
                       createTable = function(mySurveyUid, myTableNumber) {
                           #if data was selected by samples only…all surveyUids will be bla
                       op <- NA

                       thisSurveyLabel <- (self$results[[myTableNumber]] %>%
                         pull(surveyLabel))[[1]]
```

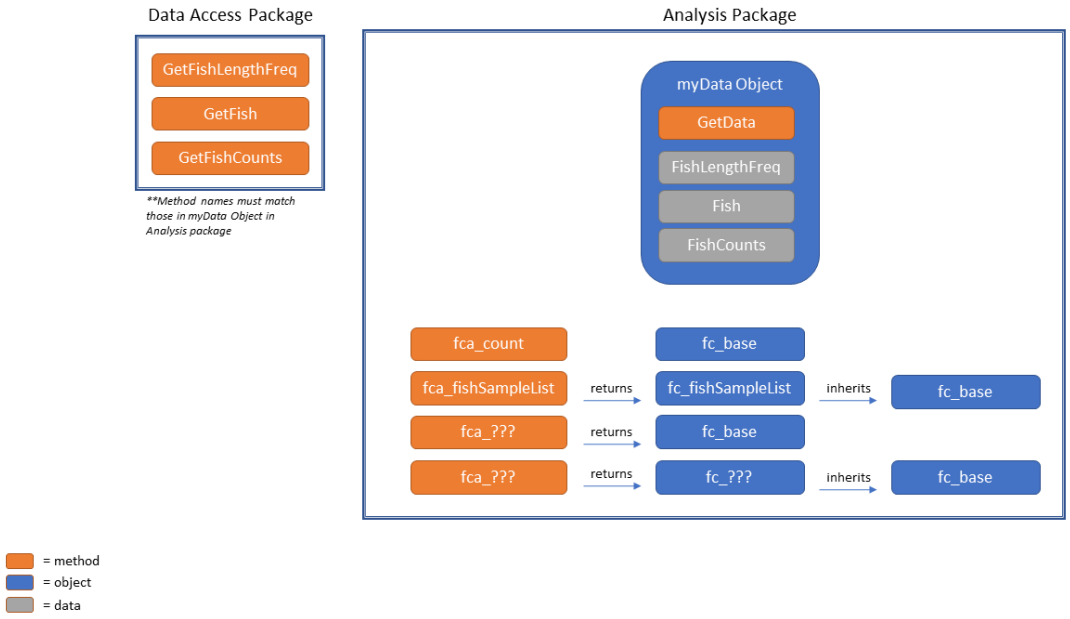
```
    op <- self$results[[myTableNumber]] %>%
      filter(surveyUid == mySurveyUid) %>%
      group_by(countSpeciesLabel, sampleLabel) %>%
      summarise(FishCount = sum(FishCount, na.rm = TRUE)) %>%
      mutate(countGenderLabel = "Total") %>%
      bind_rows(self$results[[myTableNumber]]  %>%
      filter(surveyUid == mySurveyUid)) %>%
      arrange(countSpeciesLabel, sampleLabel, countGenderLabel) %>%
      select(-countSpeciesCode,-countGenderCode,-sampleUid,-surveyLabel,
      spread(countGenderLabel, FishCount, fill = 0) %>%
      relocate("Total", .after = last_col()) %>%
      gt(rowname_col = colnames(self$results[[1]])[8]) %>%
        tab_options(
          row_group.background.color = self$groupHeaderBackgroundColor,
          summary_row.background.color = self$groupSummaryBackgroundColo
          ) %>%
        tab_header(title = md(self$tableTitle),
                   subtitle = thisSurveyLabel) %>%
        summary_rows(groups = TRUE,
                     columns = everything(),
                     fns = list("Total" = "sum"),
                     formatter = fmt_integer,
                     use_seps = TRUE,
                     missing_text = "") %>%
        self$gtTheme()

      return(op)
    }
  ))
```

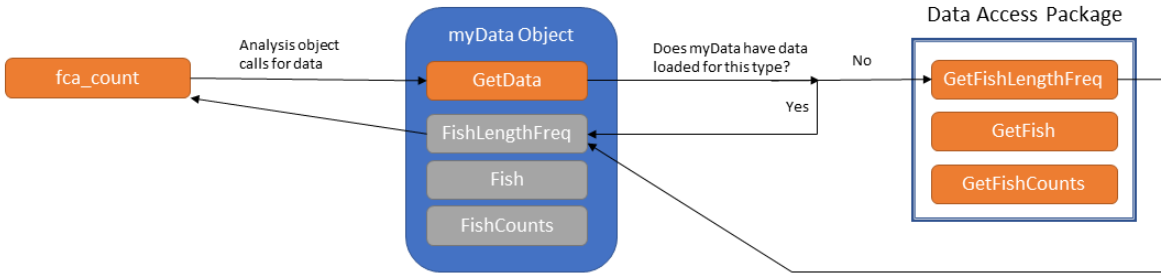
Part III

Data Access R Package

Packages



WorkFlow



Principles

By Using myData Object:

- Only data that's needed is loaded

- Data is cached
- Data is only retrieved once for entire analysis string

By Using Separate Data Access Package:

- Isolates and generalizes data access
- Allows analysis code to use different sources of data
- Allows different scripts to use different authentications for data access

Part IV

References