Python Module Index/stryke

[file:///J:/868/022/Calcs/validation/stryke/build/html/stryke.html#module-stryke](J://868/022/Calcs/validation/stryke/build/html/stryke.html#module-stryke)

Stryke: Kleinschmidt Associates Turbine Blade Strike Simulation Model

The intent of Stryke is to model downstream the passage mortality through a theoretical hydroelectric facility. The simulation employs Monte Carlo methods within an individual based modeling (IBM) framework, meaning we are modeling the individual fates of a theoretical population of fish and summarizing the results for a single simulation. Then, we iterate that IBM thousands of times and eventually we have a pretty good estimate of what the overall downstream passage survival would be of a theoretical population of fish through a hydroelectric facility.

For fish passing via entrainment, individuals are exposed to turbine strike, which is modeled with the Franke et al. 1997 equations. For fish that pass via passage structures or spill, mortality is assessed with a roll of the dice using survival metrics determined a priori or sourced from similar studies.

stryke.**enable\_matplotlib\_inline**()

*class*stryke.**epri**(*states=None*, *plant\_cap=None*, *Month=None*, *Family=None*, *Genus=None*, *Species=None*, *HUC02=None*, *HUC04=None*, *HUC06=None*, *HUC08=None*, *NIDID=None*, *River=None*)

Bases: **object**

A Python class for querying the Electric Power Research Institute (EPRI) entrainment database and fitting a distribution to the observed entrainment rates. The class allows for flexible querying of the database based on various ecological and hydrological parameters.

The class supports analysis by state, turbine capacity, date, taxonomy, feeding guild, habitat preference, water body type, and more, to explore entrainment patterns across different environmental and operational conditions.

**ExtremeFit**()

Fits an extreme value distribution to the filtered EPRI dataset to model entrainment rates. This method provides an analysis of entrainment patterns based on the shape, location, and scale parameters of the distribution.

Outputs detailed statistics of the fitted Pareto distribution, including the mean, variance, and standard deviation, offering insights into the entrainment rates’ distribution characteristics within the selected dataset.

**GumbelFit**()

Fits a Gumbel distribution to the filtered EPRI dataset to model entrainment rates. This method provides an analysis of entrainment patterns based on the shape, location, and scale parameters of the distribution.

Outputs detailed statistics of the fitted Pareto distribution, including the mean, variance, and standard deviation, offering insights into the entrainment rates’ distribution characteristics within the selected dataset.

**LengthSummary**()

Summarizes fish lengths from the filtered EPRI dataset, aggregating counts across specified length cohorts and fitting the aggregated lengths to a log-normal distribution.

The method samples uniformly within each size cohort to approximate the distribution of lengths and then fits a log-normal distribution to these sampled lengths, providing an analysis of the size distribution of the entrained fish in the selected dataset.

**LogNormalFit**()

Fits a log-normal distribution to the filtered EPRI dataset to model entrainment rates. This method provides an analysis of entrainment patterns based on the shape, location, and scale parameters of the distribution.

Outputs detailed statistics of the fitted Pareto distribution, including the mean, variance, and standard deviation, offering insights into the entrainment rates’ distribution characteristics within the selected dataset.

**ParetoFit**()

Fits a Pareto distribution to the filtered EPRI dataset to model entrainment rates. This method provides an analysis of entrainment patterns based on the shape, location, and scale parameters of the distribution.

Outputs detailed statistics of the fitted Pareto distribution, including the mean, variance, and standard deviation, offering insights into the entrainment rates’ distribution characteristics within the selected dataset.

**WeibullMinFit**()

Fits a Frachet distribution to the filtered EPRI dataset to model entrainment rates. This method provides an analysis of entrainment patterns based on the shape, location, and scale parameters of the distribution.

Outputs detailed statistics of the fitted Pareto distribution, including the mean, variance, and standard deviation, offering insights into the entrainment rates’ distribution characteristics within the selected dataset.

**epri**

I want to hook up stryke to the EPRI database when project loads, figure out how to do this cuz this is lame

**plot**()

**summary\_output**()

*class*stryke.**hydrologic**(*nid\_near\_gage\_dir*, *output\_dir*)

Bases: **object**

A Python class for conducting flow exceedance analysis using recent USGS data in relation to the contributing watershed size. It utilizes a linear relationship between drainage area and flow exceedances derived from the 100 nearest USGS gages to a given dam. This relationship is used to predict flow conditions such as a wet spring.

The class is designed to import relevant data, including dam locations and nearby USGS gages, and to calculate flow exceedances based on this data.

**curve\_fit**(*season*, *dam*, *exceedance*)

Performs an ordinary least squares (OLS) regression to model the relationship between watershed size and flow exceedance probability for a specific dam and season. This method uses the statsmodels library to fit the model and provides a summary of the regression results.

Parameters include:

* season (str) - the season for which the analysis is conducted (e.g., ‘Spring’).
* dam (str) - the identifier for the dam of interest, corresponding to entries in the NID\_to\_gage DataFrame.
* exceedance (str) - specifies the exceedance threshold being modeled (e.g., ‘exc\_90’ for the 90th percentile exceedance flow).

The function retrieves the relevant dam data, filters the exceedance DataFrame for the specified season and exceedance threshold, and extracts the drainage area and flow data for regression analysis.

Outputs: - Prints a summary of the regression model, including the p-value and model

coefficients. If the model is statistically significant (p-value < 0.05), it calculates and prints the exceedance flow for the specified dam based on its drainage area.

* Updates class attributes with the regression data (X, Y), the predicted exceedance flow for the specified dam (DamY), and the dam’s drainage area (DamX) for potential further analysis.

**seasonal\_exceedance**(*seasonal\_dict*, *exceedence*, *HUC=None*)

Calculates seasonal flow exceedance probabilities for each gage and compiles the results into a DataFrame. The method considers predefined seasons and specific exceedance thresholds to evaluate flow rates.

Parameters: - seasonal\_dict (dict): A dictionary mapping seasons to lists of month numbers,

defining the temporal scope of each season.

* exceedence (list): A list of exceedance thresholds (percentages) for which flow rates are calculated.
* HUC (int, optional): Hydrologic Unit Code to filter the analysis to a specific watershed or sub-watershed. If provided, only gages within the specified HUC are considered.

The method iterates through the gages, applying the seasonal definitions and exceedance thresholds to calculate minimum flow rates that are exceeded at the specified percentages of time within each season. Results include gage identifiers, names, HUCs, drainage areas, seasons, calculated flow rates, and exceedance percentages.

Outputs: - Updates the *exceedance* attribute of the class instance, storing the compiled

exceedance data as a DataFrame.

*class*stryke.**simulation**(*proj\_dir*, *wks*, *output\_name*, *existing=False*)

Bases: **object**

Python class object that initiates, runs, and holds data for a facility specific simulation.

**Francis**(*param\_dict*)

Calculates the survival probability of a fish passing through a Francis turbine, based on the blade strike model developed by Franke et al. 1997. This model incorporates various turbine and biological parameters to estimate the likelihood of a fish surviving after potential blade strikes within a Francis turbine.

Parameters: - length (float): Length of the fish in feet. This parameter is crucial as it

directly influences the probability of blade strike.

* param\_dict (dict): A dictionary of turbine and operational parameters required for the survival probability calculation. The expected keys and their descriptions are as follows:
  + ‘H’: Net head of water across the turbine (in feet).
  + ‘RPM’: Rotational speed of the turbine runner (in revolutions per minute).
  + ‘D’: Diameter of the turbine runner (in feet).
  + ‘Q’: Volumetric flow rate through the turbine (in cubic feet per second).
  + ‘Qper’: Percentage of the optimum flow rate, contributing to the relative flow angle calculation.
  + ‘ada’: Efficiency of the turbine, a factor in calculating the tangential flow angle.
  + ‘N’: Number of blades on the turbine.
  + ‘iota’: Ratio between discharge with no exit swirl and optimum discharge, used in calculating the relative flow angle (β).
  + ‘D1’: Diameter at the turbine entrance (in feet).
  + ‘D2’: Diameter at the turbine exit (in feet).
  + ‘B’: Width of a turbine blade (in feet).
  + ‘\_lambda’: Empirical constant for blade strike probability, recommended by the U.S. Fish and Wildlife Service.

The function first calculates the energy and discharge coefficients of the turbine, followed by the relative flow angle (β) and the tangential flow angle upstream of the runner (α). Utilizing these angles, it estimates the probability of a fish striking a blade. The survival probability is then derived as the complement of the strike probability.

Returns: - float: The probability of a fish surviving a blade strike in a Francis turbine,

ranging from 0 (no survival) to 1 (certain survival).

Example:

param\_dict = {‘H’: 100, ‘RPM’: 100, ‘D’: 10, ‘Q’: 500, ‘Qper’: 1.1, ‘ada’: 0.8,

‘N’: 4, ‘iota’: 1.1, ‘D1’: 5, ‘D2’: 10, ‘B’: 1, ‘\_lambda’: 0.2}

fish\_length = 1 # 1 foot survival\_probability = Francis(fish\_length, param\_dict) print(f’Survival Probability: {survival\_probability}’)

**Kaplan**(*param\_dict*)

Calculates the probability of a fish surviving a blade strike in a Kaplan turbine based on the model proposed by Franke et al. 1997.

The function considers various turbine parameters and the length of the fish to compute the survival probability. It employs a blend of deterministic and stochastic elements to model the interaction between fish and turbine blades, particularly considering the location of the strike along the blade.

Parameters: - length (float): The length of the fish in meters. - param\_dict (dict): A dictionary containing key turbine parameters necessary for the calculation. Expected keys and their

meanings are as follows:

* ‘H’: The net head of water across the turbine (m).
* ‘RPM’: The revolutions per minute of the turbine.
* ‘D’: The diameter of the turbine (m).
* ‘Q’: The flow rate through the turbine (m^3/s).
* ‘ada’: The specific speed of the turbine.
* ‘N’: The number of blades on the turbine.
* ‘\_lambda’: The empirically derived constant for blade strike probability,

suggested by the U.S. Fish and Wildlife Service.

The function first computes the energy coefficient (Ewd) and discharge coefficient (Qwd) of the turbine, followed by the angle of absolute flow to the axis of rotation. Utilizing these parameters, it estimates the probability of a fish striking a blade and subsequently computes the survival probability by subtracting the strike probability from one.

Returns: - The probability of a fish surviving a blade strike in a Kaplan turbine, ranging from 0 (no survival) to 1 (certain survival).

Note: The function assumes the position of the fish strike along the blade (rR) is uniformly distributed between 0.3 and 1.0, based on recommendations from Deng et al. (please refer to the specific study for more details).

Example:

param\_dict = {‘H’: 30, ‘RPM’: 100, ‘D’: 5, ‘Q’: 200, ‘ada’: 0.8, ‘N’: 4, ‘\_lambda’: 0.2} fish\_length = 0.5 # 50 cm survival\_probability = Kaplan(fish\_length, param\_dict) print(f’Survival Probability: {survival\_probability}’)

**Propeller**(*param\_dict*)

Estimates the survival probability of a fish passing through a propeller turbine, adapting the blade strike model from Franke et al. 1997. This function considers the physical characteristics of the fish and turbine operational parameters to calculate the likelihood of fish survival after potential blade strikes.

The calculation assumes a fixed position on the blade where the fish strike might occur and incorporates both deterministic and empirical parameters to model the interaction dynamics.

Parameters: - length (float): The length of the fish in meters, which is a critical factor in determining the probability of a blade strike. - param\_dict (dict): A dictionary containing essential turbine parameters for the calculation. Expected keys include:

* ‘H’: The net head of water across the turbine (m).
* ‘RPM’: The revolutions per minute of the turbine.
* ‘D’: The diameter of the turbine (m).
* ‘Q’: The actual flow rate through the turbine (m^3/s).
* ‘ada’: The specific speed of the turbine.
* ‘N’: The number of blades on the turbine.
* ‘Qopt’: The optimum flow rate through the turbine (m^3/s), used in

beta angle calculations. - ‘Qper’: The percentage of the optimum flow rate, contributing to the beta calculation. - ‘\_lambda’: An empirical constant for blade strike probability, suggested by the U.S. Fish and Wildlife Service.

The function computes the energy coefficient (Ewd) and discharge coefficient (Qwd) to determine the turbine’s operational state, followed by the calculation of the absolute flow angle to the rotation axis (a\_a). The strike probability is then estimated using the fish length, turbine diameter, and the calculated flow angle, with the survival probability being the complement of the strike probability.

Returns: - The probability of a fish surviving a blade strike in a propeller turbine, with values ranging from 0 (no survival) to 1 (certain survival).

Note: The function assumes a fixed radial position (rR = 0.75) for the fish strike on the blade, simplifying the model while maintaining relevance to typical turbine conditions. This decision is based on empirical observations and simplifies the model’s complexity.

Example:

param\_dict = {‘H’: 25, ‘RPM’: 120, ‘D’: 3, ‘Q’: 180, ‘ada’: 0.7, ‘N’: 3,

‘Qopt’: 160, ‘Qper’: 0.8, ‘\_lambda’: 0.2}

fish\_length = 0.4 # 40 cm survival\_probability = Propeller(fish\_length, param\_dict) print(f’Survival Probability: {survival\_probability}’)

**Pump**(*param\_dict*)

Pump mode calculations from fish entrainment analysis report: J:Q0DocsEntrainmentEntrainment CalcsBladeStrike\_CabotStation.xlsx

**create\_hydrograph**(*discharge\_type*, *scen*, *scen\_months*, *flow\_scenarios\_df*, *fixed\_discharge=None*)

Generates a hydrograph for simulation based on specified scenarios, either by importing and adjusting actual hydrograph data or by simulating a hydrograph with a fixed discharge rate across specified months.

Parameters: - discharge\_type (str): Specifies the type of discharge data to use; can be

‘hydrograph’ for actual hydrograph data or ‘fixed’ for a constant discharge rate.

* scen (str): The scenario identifier to filter the relevant data in the flow scenarios dataframe.
* scen\_months (list): List of integers representing the months to include in the hydrograph simulation.
* flow\_scenarios\_df (DataFrame): A dataframe containing flow scenario data, including gage information, proration factors, and flow years for ‘hydrograph’ type scenarios.
* fixed\_discharge (float, optional): The fixed discharge rate (in cubic feet per second) to use for ‘fixed’ discharge type simulations.

Returns: - DataFrame: A pandas DataFrame representing the hydrograph for the specified

scenario, containing columns for dates, prorated daily average flow, and month.

The function handles two main types of simulations: importing and adjusting USGS hydrograph data based on a proration factor or simulating a hydrograph with a constant discharge rate across the specified months.

**create\_route**(*wks\_dir*)

Constructs a directed graph representing the migratory network of fish, using data from nodes and edges defined within a specified project database. The graph is built using the NetworkX library, allowing for the utilization of its extensive graph analysis functionalities.

The method reads node and edge information from an Excel file located in the provided directory, ‘wks\_dir’. Nodes represent points within the migratory network (such as entry, exit, and decision points), while edges represent the possible paths a fish can take between these nodes, along with associated weights (e.g., probabilities or costs associated with each path).

Parameters: - wks\_dir (str): The directory path where the project database Excel file is

located. This file should contain ‘Nodes’ and ‘Edges’ sheets with the necessary data to construct the network graph.

The method performs the following steps: 1. Reads the ‘Nodes’ and ‘Edges’ sheets from the Excel file, extracting relevant

data for graph construction.

1. Initializes an empty directed graph (DiGraph) object using NetworkX.
2. Adds nodes to the graph based on locations specified in the ‘Nodes’ data.
3. Iterates over the ‘Edges’ data to add directed edges between nodes, assigning weights to these edges as specified in the data.
4. Calculates the maximum number of moves a fish can make within the network, based on all shortest paths from a predefined start node (‘river\_node\_0’) to the furthest node, identified by ‘self.max\_river\_node’.

The constructed graph and the maximum number of moves are stored in ‘self.graph’ and ‘self.moves’, respectively, for use in subsequent analyses or simulations within the broader context of the project.

Note: This method assumes the presence of a start node labeled ‘river\_node\_0’ and relies on ‘self.max\_river\_node’ being set prior to its call to determine the endpoint for path calculations.

**daily\_hours**(*ops\_df*, *q\_cap\_dict*, *op\_order\_dict*, *operations='independent'*)

Simulates the daily operational hours of units in hydroelectric facilities, considering the facility type (run-of-river or peaking) and operational dependencies between units. The function adjusts flow rates based on operational hours and unit capacities.

Parameters: - ops\_df (DataFrame): Contains unit operation parameters, including potential

log-normal distribution parameters for operation hours and probability of not operating.

* q\_cap\_dict (dict): Maps each unit to its flow capacity (cubic feet per second).
* op\_order\_dict (dict): Maps each unit to its operational order, important for facilities with dependent unit operations (e.g., first on, last off considerations).
* operations (str, optional): Defines the operational mode of the units; can be ‘independent’ for units operating independently or ‘dependent’ for units with sequential dependencies. Defaults to ‘independent’.

Returns: - tuple: Contains total operational hours, total flow, a dictionary of operational

hours per unit, and a dictionary of flow per unit.

The function supports various operational scenarios, including fixed hours, hours determined by a log-normal distribution, or dependent operations where the operation of one unit depends on another. For run-of-river facilities, units operate 24/7, while peaking facilities may vary.

**get\_USGS\_hydrograph**(*gage*, *prorate*, *flow\_year*)

Retrieves and standardizes a hydrograph from the USGS for a specified flow year, and adjusts the flow based on watershed size. The function fetches gage data online, extracts relevant flow data, and applies a proration factor to account for differences in watershed size.

Parameters: - gage (str): The USGS gage identifier for which the hydrograph is requested. - prorate (float): Factor by which to adjust the flow data, typically based on

the relative size of the watershed of interest compared to that of the gage.

* flow\_year (int): The year for which flow data is to be retrieved and processed.

Returns: - DataFrame: A pandas DataFrame containing the daily average flow data for the

specified flow year, adjusted by the proration factor. The DataFrame includes columns for date, original daily average flow (‘DAvgFlow’), prorated daily average flow (‘DAvgFlow\_prorate’), year, and month.

Note: The function assumes the availability of USGS gage data online and may require internet access to fetch the data. The use of ‘verify=False’ in the requests.get call may lead to security warnings and should be used with caution.

**movement**(*location*, *status*, *swim\_speed*, *graph*, *intake\_vel\_dict*, *Q\_dict*, *op\_order*)

Simulates the movement of a fish through a hydroelectric project’s infrastructure, considering operational conditions, the fish’s swimming capabilities, and environmental requirements. It calculates the probability of the fish choosing various paths based on water discharge and unit status.

Parameters: - location (str): Current location within the project infrastructure. - status (int): Survival status of the fish, 1 for alive and 0 for dead. - swim\_speed (float): Swimming speed of the fish, for intake velocity resistance. - graph (networkx.Graph): Directed graph of the project, with nodes as locations

and edges as paths between locations.

* intake\_vel\_dict (dict): Maps each turbine to its intake velocity.
* Q\_dict (dict): Contains discharge information, including ‘curr\_Q’ for current discharge, ‘min\_Q’ for minimum operating discharge, ‘env\_Q’ for environmental flow, ‘sta\_cap’ for station capacity, and turbine capacities.
* op\_order (dict): Maps each turbine to its operational order, for sequence determination as discharge increases.

Special logic is applied in the forebay area, considering the variety of paths (turbines vs. spillway) and operational conditions. Movement probabilities are calculated based on edge weights, apportioned by operational conditions and the fish’s swimming abilities.

Returns: - str: New location after movement simulation, which could be a specific turbine,

‘spill’ for the spillway, or the same location if the fish cannot move.

This method provides a detailed simulation of fish movement, incorporating environmental flows, operational priorities, and biological capabilities to inform management decisions and impact assessments within hydroelectric projects.

**node\_surv\_rate**(*length*, *status*, *surv\_fun*, *route*, *surv\_dict*, *u\_param\_dict*)

Calculates the survival probability of a fish passing through a node in the migratory network, taking into account the type of hydraulic structure encountered (e.g., Kaplan, Propeller, Francis turbines, or pump mode operation) and the fish’s length. The method supports both predefined survival probabilities (a priori values) and dynamic calculations based on turbine parameters.

Parameters: - length (float): The length of the fish, which is a critical factor in calculating the blade strike probability. - status (int): The status of the fish, where 0 indicates a condition (such as mortality) that precludes further survival calculation. - surv\_fun (str): The survival function or method to be used. This can be ‘a priori’ for predefined survival probabilities, or the name of a specific turbine type (‘Kaplan’, ‘Propeller’, ‘Francis’) for dynamic survival probability calculation. ‘Pump’ mode is also supported for scenarios involving fish entrainment during pumping operations. - route (str): The migratory route or node identifier. In the case of ‘a priori’ survival probabilities, this corresponds to the key in the ‘surv\_dict’ dictionary. For dynamic calculations, it specifies the turbine for which parameters are provided in ‘u\_param\_dict’. - surv\_dict (dict): A dictionary containing predefined survival probabilities for various routes or nodes, used when ‘surv\_fun’ is set to ‘a priori’. The keys correspond to route identifiers, and the values are the survival probabilities. - u\_param\_dict (dict): A dictionary of dictionaries containing turbine parameters necessary for calculating survival probabilities. Each key corresponds to a route or turbine identifier, with the associated value being another dictionary of parameters relevant to the specific turbine type indicated by ‘surv\_fun’.

Returns: - float: The calculated survival probability for the fish at the given node, converted to a 32-bit float. If the ‘status’ parameter is 0, indicating a non-survivable condition, the method immediately returns 0.0.

This method integrates various survival probability models, including those based on empirical data (‘a priori’) and dynamic models for different turbine types, providing a versatile tool for assessing fish survival in hydroelectric project simulations.

**population\_sim**(*spc\_df*, *discharge\_type*, *tot\_hours*, *tot\_flow*, *curr\_Q*)

Simulates a population of fish based on species-specific parameters and operational conditions of a hydroelectric facility. The function uses distribution parameters to simulate entrainment rates, which are then adjusted based on operational conditions and historical data.

Parameters: - spc\_df (DataFrame): Contains species-specific parameters for simulating

entrainment rates, including distribution type and parameters.

* discharge\_type (str): Type of discharge operation (‘fixed’ or variable).
* tot\_hours (float): Total operational hours of the facility.
* tot\_flow (float): Total flow through the facility in cubic feet.
* curr\_Q (float): Current discharge rate in cubic feet per second.

Returns: - int: Simulated number of fish entrained, rounded to the nearest whole number.

The simulation considers the discharge type, operational hours, and current flow to estimate the volume of water interacting with the fish population. The entrainment rate is drawn from the specified distribution and adjusted for feasibility based on historical data.

**run**()

Executes a comprehensive simulation of fish populations navigating through a hydroelectric facility, accounting for various operational scenarios, species-specific behaviors, and environmental conditions. The function integrates multiple components including route creation, operational parameters setup, and survival probability calculations.

The simulation workflow includes: 1. Route creation for fish movement based on facility layout. 2. Initialization of dictionaries for operational parameters, survival

probabilities, intake velocities, unit capacities, and operational orders.

1. Iteration over hydroelectric units to populate dictionaries with unit- specific operational data.
2. Scenario-based simulation, considering different flow conditions and species presence.
3. Population simulation for each species under each scenario, including entrainment rate calculations and survival assessments.
4. Movement and survival simulation for individual fish, with results stored in a hierarchical data format (HDF) file for analysis.

No parameters are passed directly to this function as it operates on the class attributes set during initialization and updated through other methods.

The function logs progress and notable events (e.g., entrainment events, units not operating) throughout the simulation, providing insights into the simulation dynamics and outcomes.

Outputs: The function generates and stores detailed simulation results in an HDF file, including daily summaries, entrainment statistics, and length distributions for simulated fish populations across different scenarios.

**speed**(*A*, *M*)

Calculates swimming speed based on fish length, caudal fin aspect ratio, and swimming mode, according to Sambilay 1990. The function is vectorizable, allowing for efficient calculations over arrays of inputs.

Parameters: - L (float or array-like): Fish length in centimeters. - A (float or array-like): Caudal fin aspect ratio (dimensionless), obtained

from fish morphological data.

* M (int or array-like): Swimming mode, where 0 represents sustained swimming and 1 represents burst swimming.

Returns: - float or array-like: Swimming speed in kilometers per hour. The conversion

factor used (0.911344) translates the speed to feet per second for compatibility with certain applications.

This model provides an estimate of swimming speeds for fish based on physical characteristics and behavior, useful in ecological and engineering studies related to aquatic locomotion and habitat interactions.

**summary**()

Summarizes the results of fish entrainment simulations stored in an HDF file. This function aggregates and analyzes data across species, scenarios, and units to provide insights into entrainment risks and outcomes.

The summary process includes: 1. Accessing the HDF store and retrieving relevant data tables (population,

flow scenarios, unit parameters, daily summaries, and length distributions).

1. Iterating through species and scenarios to calculate survival rates, length statistics, and entrainment outcomes.
2. Fitting beta distributions to survival probabilities and summarizing the results.
3. Aggregating daily data to provide cumulative summaries, including median population sizes, entrainment and mortality rates, and confidence intervals.
4. Analyzing entrainment and mortality distributions to calculate probabilities of exceeding certain thresholds (e.g., 10, 100, 1000 individuals).

Outputs: - Detailed summaries are printed to the console, providing an overview of

simulation outcomes.

* Key summary statistics, including beta distribution parameters and cumulative sums, are stored as class attributes for further analysis or reporting.
* The function updates the HDF file with aggregated summary data and closes the file upon completion.

Note: This function assumes that the HDF file contains the necessary data tables from previous simulation runs and that the file structure adheres to the expected format.