Supporting information: Using Survival Analysis to Develop Models for Estimating Size-at-Detection of Invasive Species under Surveillance

This document serves as a concise user guide for the R analysis program introduced in the study "Using Survival Analysis to Develop Models for Estimating Size-at-Detection of Invasive Species under Surveillance." We present the functional R program named "**select_function**", which generates various mathematical functions—including linear, quadratic, exponential, and logistic—based on user inputs. The program then calculates the integral of the chosen function to verify its validity (ensuring it is close to 1) and, if valid, plots the function. Below is a detailed breakdown to help users understand and utilize each step effectively:

```
select_function <- function(type = c("linear","quadratic", "exponential", "logistic"),
                                        params = list(n = n, p = p, x0 = x0, M = M, r = r)) {

  # Match the selected type to ensure it is one of the allowed values.
  type <- match.arg(type)

  # Define the specific function based on the type parameter.

   fun <- switch(type,

# Linear function calculation
      linear = function(y) {
   # Check if the required parameters n and p are provided.
        if (is.null(params$n) || is.null(params$p))
          stop("n and p parameters are required for custom linear function")

   # Calculate h based on the provided parameters and the input value y.
        h <- 1 - (1 - params$p)^(params$n * y)
        return(h * exp(-h / (params$n * log(1 - params$p)) - y))
      },

   # Quadratic function calculation
      quadratic = function(y) {
        if (is.null(params$n) || is.null(params$p))
          stop("n and p parameters are required for custom quadratic function")
        h <- 1 - (1 - params$p)^(params$n * y^2)
```

```
        return(h * exp(-h / (params$n * log(1 - params$p)) - y^2) * 2 * y)
      },

  # Exponential function calculation
      exponential = function(y) {
        if (is.null(params$n) || is.null(params$p) || is.null(params$x0) || is.null(params$r))
          stop("n, p, x0 and r parameters are required for exponential function")
        exp_term <- params$x0 * exp(params$r * y)
        h <- 1 - (1 - params$p)^(params$n * exp_term)
        return(h * exp(-h / (params$n * log(1 - params$p)) - exp_term) * params$r *
exp_term)
      },

  # Logistic function calculation
      logistic = function(y) {
        if (is.null(params$n) || is.null(params$p) || is.null(params$x0) || is.null(params$M)
|| is.null(params$r))
          stop("n, p, x0, r and M parameters are required for logistic function")
        logistic_term <- params$M / (1 + ((params$M - params$x0)/params$x0) * exp(-
params$r * y))
        h <- 1 - (1 - params$p)^(params$n * logistic_term)
        return(h * exp(-h / (params$n * log(1 - params$p)) - logistic_term) * params$r *
logistic_term * (1 - logistic_term/params$M))
      }
    )

    # Calculate the integral of the function using numerical integration.
    integral <- tryCatch({
      integrate(fun, lower = 0, upper = 1000)$value
    }, error = function(e) {
      warning("Error in integration: ", e$message)
      return(NA)
    })

     # Check if the calculated integral is close to 1 (indicating a valid probability
distribution).
    is_valid <- abs(integral - 1) < 1e-4
    # Calculate mean and q90 if the function is valid
```

```
  mean_val <- NA
  q90_val <- NA
  if (is_valid) {
    # Calculate mean
    mean_fun <- function(y) y * fun(y)
    mean_val <- tryCatch({
      integrate(mean_fun, lower = 0, upper = 1000)$value
    }, error = function(e) {
      warning("Error in mean calculation: ", e$message)
      return(NA)
    })

    # Calculate q90
    cdf_fun <- function(y) {
      tryCatch({
        integrate(fun, lower = 0, upper = y)$value
      }, error = function(e) {
        warning("Error in CDF calculation: ", e$message)
        return(NA)
      })
    }
    q90_val <- uniroot(function(y) cdf_fun(y) - 0.9, c(0, 1000))$root
  }

  # If valid, create a density plot
  plot <- NULL
  if (is_valid) {
    x <- seq(0, 150, length.out = 1000)
    y <- sapply(x, fun)
    df <- data.frame(x = x, y = y)
    # Conditionally format the parameters based on the function type
    param_text <- switch(type,
      linear = bquote(italic(n) == .(params$n) * "," ~ italic(p) == .(params$p)),
      quadratic = bquote(italic(n) == .(params$n) * "," ~ italic(p) == .(params$p)),
      exponential = bquote(italic(n) == .(params$n) * "," ~ italic(p) == .(params$p) * "," ~ italic(x)[0] == .(params$x0) * "," ~ italic(r) == .(params$r)),
      logistic = bquote(italic(n) == .(params$n) * "," ~ italic(p) == .(params$p) * "," ~ italic(x)[0] == .(params$x0) * "," ~ italic(M) == .(params$M) * "," ~ italic(r)
```

```
== .(params$r))
  )
    # Title text with the parameter and q90 formatting
    title_text <- bquote(atop(
      .(type),
      atop(
        .(param_text),
        paste("Mean = ", .(sprintf("%.2f", mean_val)), ", ", italic(q)[90], " =
", .(sprintf("%.2f", q90_val)))
      )
    ))
    plot <- plot(df, ylab = bquote(italic(f(t))), xlab = bquote(italic(t)), type = "l", lwd = 2,
main = title_text,cex.main=1.5)
  }

  # return the results
  return(list(
    fun = fun,
    integral = integral,
    is_valid = is_valid,
    plot = plot,
    mean = mean_val,
    q90 = q90_val
  ))
}
# Helper function to display results
plot_function <- function(type, params) {
  result <- select_function(type, params)
  if (result$is_valid) {
    print(result$plot)
    cat("Integral:", result$integral, "\n")
    cat("Is valid:", result$is_valid, "\n")
    cat("Mean:", result$mean, "\n")
    cat("90th percentile:", result$q90, "\n")
  } else {
    cat("The function is not valid. Integral:", result$integral, "\n")
  }
}
```

Arguments of the "select_function" program:

type This function returns a mathematical function based on the specified type including linear, quadratic, exponential, or logistic.

params It requires various parameters such as n, p, x_0, M, and r depending on the function type.

The result from the "select_function" program, which is a list with the following components:

Integral the value of the integral

Is valid* TRUE if valid, that is the calculated integral is close to 1 (indicating a valid probability distribution).

Mean mean of the random variable

90th percentile 90th percentile of the random variable

Examples:

```
plot_function("linear", list(n = 200, p = 0.001))
plot_function("quadratic", list(n = 200, p = 0.1))
plot_function("exponential", list(n = 400, p = 0.001, x0 = 0.001, r = 0.099))
plot_function("logistic", list(n = 10, p = 0.001, x0 = 0.001, M = 50, r = 0.05))
```

* The relationship between size and time is governed by a distinct function. In rare cases, such as with a logistic model, converting size to time without accounting for all time values restricts the range of integration. As a result, the calculated integral over time is less than one.