Arithmetic Functions

Type	Function	Description	Example
Rounding	round	Round to given precision	round(my_var, digits = 3)
Minimum	pmin	Get minimum of two or more values	pmin(var_1, var_2)
Maximum	pmax	Get maximum of two or more values	pmax(var_1, var2)
Logarithm	log	Take logarithm with given base. Base may be optionally specified (default is base e)	log(my_var) log(my_var, base = 2)
Floor	floor	Round down to nearest integer	floor(my_var)
Ceiling	ceiling	Round up to nearest integer	ceiling(my_var)
Exponentiation	exp	Raise base to given power. Base may be optionally specified (default is base e)	exp(4.3) exp(my_var, base = 10)

Arithmetic Operators

Type	Operator	Example
Addition	+	20 + 15
Subtraction	-	A - B
Multiplication	*	23 * 4
Division	/	30 /2
Exponent	^	100^3
Integer Division	%/%	17 %/% 3

Lookup & Other Functions

Function	Description	Example
look_up	Look up a value in a table. Similar to *vlookup* in Excel. By default, look_up will only perform exact matching. To perform a range lookup, set optional argument "bin" equal to TRUE.	look_up(my_table, "my_column" = 10, value = "value_column")
rescale_prob	Rescale a probability defined over a given interval to one defined over a different interval, assuming a constant rate. Used to define transition probabilities irrespective of cycle length.	rescale_prob(my_prob, from = 30, to = 1)
rate_to_prob	Calculate probability of an event given a rate, length of the interval, and duration over which rate is estimated.	rate_to_prob(my_rate, to = 365, per = 30)
dispatch_strategy	Make a formula whose value depends on the strategy. Expects one argument for each strategy with name of the argument corresponding to strategy name and value corresponding to value for that strategy.	dispatch_strategy ("strategy_a" = 10, "strategy_b" = 30,"strategy_c" = 100)

hēRo3 Keywords

Name	Description
bc	Base-case value for a given parameter. This keyword can *only* be used on the *DSA Inputs* and *PSA Inputs* pages.
C	Calculates complementary probability in a transition matrix or initial state probability. Can only be used on the Transitions and States pages.
cycle_length_days	Length of a model cycle, in days.
cycle_length_weeks	Length of a model cycle, in weeks.
cycle_length_months	Length of a model cycle, in months.
cycle_length_years	Length of a model cycle, in years.
group	Name of a patient group. Can be used in conditional statements to make a formula depend on group to which a patient belongs.
model_time	Number of cycles since start of a model. Counting begins with 1 in first cycle.
model_day	Time since start of model, in days. Is equal to model_time * cycle_length_days.
model_week	Time since start of model, in weeks. Is equal to model_time * cycle_length_weeks.
model_month	Time since start of model, in months. Is equal to model_time * cycle_length_months.
model_year	Time since start of model, in years. Is equal to model_time * cycle_length_years.
state_time	Number of cycles since entry into a given state. Counting begins with 1 for first cycle in the state. Can only be used in Markov Cohort models.
state_day	Time since entry into a state, in days. Equal to state_time * cycle_length_days. Can only be used in Markov Cohort models.
state_week	Time since entry into a state, in weeks. Equal to state_time * cycle_length_weeks. Can only be used in Markov Cohort models.
state_month	Time since entry into a state, in months. Equal to state_time * cycle_length_months. Can only be used in Markov Cohort models.
state_year	Time since entry into a state, in years. Equal to state_time * cycle_length_years. Can only be used in Markov Cohort models.

Survival Functions

Function	Description	Example
apply_af	Applies acceleration factor to a survival distribution.	surv_prob(my_dist, 0.65)
apply_hr	Applies hazard ratio to a survival distribution.	apply_hr(my_dist, 0.5)
apply_or	Applies odds ratio to a survival distribution.	apply_or(my_dist, 1.3)
apply_shift	Shift survival distribution forwards or backwards in time.	apply_shift(my_dist, 20)
event_prob	Calculates probability of experiencing an event in a set of intervals given a survival distribution, start times of intervals, and end times of intervals. Useful for calculating transition probabilities based either on state time or model time using a survival distribution.	event_prob(my_dist, start, end)
surv_prob	Calculates survival probabilities for survival distribution at given times.	surv_prob(my_dist, times)
add_hazards	Combine two survival distributions as two independent risks.	add_hazards(dist1, dist2)
define_survival	Define parametric survival model for given family and with given parameter values. Supported distributions include: · Exponential (exp) · Weibull (weibull) · Lognormal (lnorm) · Log-Logistic (llogis) · Gompertz (gompertz) · Generalized Gamma (gengamma) · Generalized F (genf) Information on parameterization of each survival distribution can be found here.	define_survival (dist = "weibull", shape = 1.2, scale = 40)
define_survival_cure	Define mixture or non-mixture parametric survival cure model for a given family, with given cure fraction, and with given parameter values. Supports same set of distributions and parameterizations as with "define_survival".	define_survival_cure (dist = "weibull", theta = 0.5)
define_spline_survival	Define parametric survival model for given family and with given parameter values. Supported distributions include: · Exponential (exp) · Weibull (weibull) · Lognormal (lnorm) · Log-Logistic (llogis) · Gompertz (gompertz) · Generalized Gamma (gengamma) · Generalized F (genf)	define_spline_survival (gamma1 = -2.08, gamma2 = 2.75, gamma3 = 0.23, knots1 = -1.62, knots2 = 0.57, knots3 = 1.191, (scale = 'hazard')
define_surv_table	Define survival distribution based on a table of times and survival probabilities. Columns must be named "time" and "survival" respectively.	define_surv_table(my_table)
define_surv_lifetable	Define survival distribution based on a life table. First argument is a table with annual death rates by age containing columns, "age", "male", and "female". Second and third arguments are starting age and percentage of patients who are men.	define_surv_lifetable (life_table, 50, 0.47)
join	Join two or more survival distributions together at given cut time(s). First argument is survival distribution for first segment, followed by cut point, and then distribution used in second segment.	join(km_dist, 60, param_dist)
mix	Mix two or more distributions using given weights. Each survival distribution argument must be followed by additional numeric argument specifying weight assigned to that distribution.	mix(dist1, 0.7, dist2, 0.3)
set_covariates	Takes a survival model estimated with covariates (from survfit, flexsurvreg, or other supported functions) and sets covariate values for which survival projections will be used.	set_covariates(surv_model, age = 18, prognosis = "Poor")

Logical Functions

Function	Description	Example
any()	This function evaluates if, among a set of logical vectors (ie, a table of values or a list of values or a single value), any of said vectors are TRUE. In the example to the right, if the life table includes values above 100, this function would evaluate as TRUE; otherwise, it would evaluate as FALSE.	any(Life_Table > 100)
all()	This function evaluates if, among a set of logical vectors (ie, a table of values or a list of values or a single value), all of said vectors are TRUE. In the example to the right, this function would evaluate as FALSE.	m <- c(0.1, 1.7, 2.4, 3.1, 0.8) all(m < 1)
is.na()	This function evaluates each element of a vector to determine if said element is N/A. The length of the output will be equal to the length of the input vector.	is.na()
if() else	This function evaluates a logical test and if TRUE uses the first value and if FALSE uses the second value. This function should be used if the values being returned are tables or survival distributions. The condition being checked with this function cannot be time-dependent.	if(a == "yes") Life_Table else Life_Table_2
ifelse()	This function evaluates a logical test and if TRUE uses the first value and if FALSE uses the second value. This function should be used if the values being returned are parameters or constants.	ifelse(model_time == 1, initial_cost, 0)
switch()	This function evaluates multiple checks and returns values based on those checks. Unlike either of the "if" "else" functions, this function is not limited as an either or type function and it can evaluate more than one logical test. This function may be used to determine a distribution to use based on the value of a flag with multiple possible values. Each argument name corresponds to a potential value for flag, and the argument value corresponds to the value that will be used in that case. (NOTE: flag being checked cannot be time-dependent.)	switch(dist_flag[1], "weibull" = weibull_dist, "exp" = exp_dist, "gamma" = gamma_dist)

Logical Operators

Operator	Description	Example
==	The equality operator is used to assess whether one variable, constant, or hēRo3 keyword is equal to another. If the first item is equal to the second, this will return TRUE; otherwise it will return FALSE. This operator can be used for time-dependent variables and can return different values for different cycles.	6 == model_time
!=	This operator is used to assess whether one variable, constant, or hēRo3 keyword is NOT equal to another. If the first item is NOT equal to the second, this will return TRUE; otherwise it will return FALSE. This operator can be used for time-dependent variables and can return different values for different cycles.	start_age != current_age
!	This operator returns the opposite of a logical assessment (ie, if TRUE, this operator would return FALSE). (NOTE: since R evaluates non-zero numbers as TRUE and zero as FALSE, the example to the right would return FALSE.)	!4
>=	This operator assesses if the element on the left side of the operator is greater than or equal to the element on the right side of the operator.	start_age + model_time >= menopause_age
>	This operator assesses if the element on the left side of the operator is greater than the element on the right side of the operator.	model_time > 0
<=	This operator assesses if the element on the left side of the operator is less than or equal to the element on the right side of the operator.	start_age <= current_age
<	This operator assesses if the element on the left side of the operator is less than the element on the right side of the operator.	pct_male < pct_female
&&	This operator examines only the first element and returns a single logical vector. In the example to the right, it evaluates if the first element of x and the first element of y are both TRUE or both FALSE (ie, FALSE in this example). This operator is rarely used.	x <- c(12, TRUE, FALSE) y <- c(0, TRUE,FALSE) x&&y
||	This operator examines only the first element and returns a single logical vector. In the example to the right, it evaluates if the first element of x or the first element of y is TRUE or if both are FALSE (ie, TRUE in this example). This operator is rarely used.	x <- c(12, TRUE, FALSE) y <- c(0, TRUE,FALSE) x||y
&	This operator examines each element and returns a logical vector with a length equal to that of the longer input vector. In the example to the right, it evaluates if the elements of x and the elements of y are TRUE or FALSE (ie, FALSE TRUE FALSE in this example).	x <- c(12, TRUE, FALSE) y <- c(0, TRUE,FALSE) x&y
|	This operator examines each element and returns a logical vector with a length equal to that of the longer input vector. In the example to the right, it evaluates if the elements of x or the elements of y are TRUE or FALSE (ie, TRUE TRUE FALSE in this example).	x <- c(12, TRUE, FALSE) y <- c(0, TRUE,FALSE) x|y
%in%	This operator allows you to compare vectors of different lengths (including a length of one) to see if at least one element of the first vector matches at least one element of the second vector. This can also be used to evaluate whether a value is contained within a table. The output length will equal the length of the first vector being compared (the example to the right would return TRUE).	a = 7 b <- c(12, 6, 7, 34) a%%b