Package ‘packDAMipd’

March 24, 2024

Type Package

Title Decision Analysis Modelling Package with Parameters Estimation Ability from Individual Patient Level Data

Version 1.1.0

Maintainer Sheeja Manchira Krishnan <sheejamk@gmail.com>

Description A collection of functions to construct Markov model for model-based cost-effectiveness analysis. This includes creating Markov model (both time homogenous and time dependent models), decision analysis, sensitivity analysis (deterministic and probabilistic). The package allows estimation of parameters for the Markov model from a given individual patient level data, provided the data file follows some standard data entry rules.

License GPL-3

Depends R (>= 4.3.0)

Imports readxl, stringr, data.table, reshape2, rlang, stats, lme4, survminer, SurvRegCensCov, survival, MASS, systemfit, IPDFileCheck, valueEQ5D, car, ggplot2, grDevices, lmtest, broom, effects, gvlma, methods, relaimpo, dplyr, tidyr, hash, haven, flextable, ggpubr, labelled, purrr, ISLR

Suggests knitr, rmarkdown, covr, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

URL https://github.com/sheejamk/packDAMipd

BugReports https://github.com/sheejamk/packDAMipd/issues

NeedsCompilation no

Author Sheeja Manchira Krishnan [aut, cre]

Repository CRAN

Date/Publication 2024-03-24 21:10:02 UTC
R topics documented:

- add_entries_sameuse_timepoint .................................................. 5
- adl_scoring.df ............................................................................. 6
- assign_parameters ........................................................................ 6
- blank.df ....................................................................................... 7
- calculate_icer_nmb ..................................................................... 8
- checks_markov_pick_method ............................................................... 9
- checks_plot_dsa ........................................................................... 10
- check_equal_columncontents_NAomitted ......................................... 11
- check_equal_sumcolumncontents_NAomitted ...................................... 12
- check_link_glm ............................................................................. 13
- check_list_markov_models ............................................................... 13
- check_null_na ............................................................................... 14
- check_trans_prob .......................................................................... 15
- check_treatment_arm ..................................................................... 16
- check_values_states .................................................................... 16
- combine_markov ............................................................................ 17
- combine_state .............................................................................. 18
- convert_freq_diff_basis ................................................................. 19
- convert_to_given_timeperiod ......................................................... 19
- convert_volume_basis ................................................................... 20
- convert_weight_diff_basis ............................................................... 20
- convert_wtpertimediff_basis ........................................................... 21
- convert_wtpervoldiff_basis .............................................................. 22
- costing_AandE_admission ............................................................... 22
- costing_inpatient_daycase_admission .............................................. 24
- costing_opioid_liquids_averageMED_long ....................................... 25
- costing_opioid_liquids_averageMED_wide ........................................ 26
- costing_opioid_patches_averageMED_long ....................................... 29
- costing_opioid_patches_averageMED_wide ....................................... 31
- costing_opioid_tablets_averageMED_long ....................................... 33
- costing_opioid_tablets_averageMED_wide ....................................... 35
- costing_opioid_tablets_MED_wide ................................................ 37
- costing_resource_use .................................................................... 39
- cost_data.df .................................................................................. 41
- create_new_dataset ....................................................................... 42
- create_shorttable_from_gtsummary_compare_twogroups_timpoints ....... 42
- create_table_from_gtsummary_compare_twogroups ................................ 43
- define_parameters ........................................................................ 44
- define_parameters_psa ................................................................... 45
- define_parameters_sens_anal ............................................................. 46
- define_transition_table .................................................................. 47
- do_psa ............................................................................................. 48
- do_sensitivity_analysis ................................................................. 49
- encode_codes_data ....................................................................... 50
- eval_assign_trans_prob .................................................................. 51
- eval_assign_values_states ............................................................. 52
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>find_glm_distribution</td>
</tr>
<tr>
<td>find_keyword_rand_generation</td>
</tr>
<tr>
<td>find_keyword_regression_method</td>
</tr>
<tr>
<td>find_parameters_btn_operators</td>
</tr>
<tr>
<td>find_required_parameter_comb</td>
</tr>
<tr>
<td>find_rowwise_sum_multiplecol</td>
</tr>
<tr>
<td>find_survreg_distribution</td>
</tr>
<tr>
<td>form_expression_glm</td>
</tr>
<tr>
<td>form_expression_lm</td>
</tr>
<tr>
<td>form_expression_mixed_model_lme4</td>
</tr>
<tr>
<td>generate_wt_time_units</td>
</tr>
<tr>
<td>generate_wt_vol_units</td>
</tr>
<tr>
<td>get_age_details</td>
</tr>
<tr>
<td>get_colnames_codedvalues</td>
</tr>
<tr>
<td>get_cost_AandE_code</td>
</tr>
<tr>
<td>get_cost_AandE_description</td>
</tr>
<tr>
<td>get_cost_ip_dc_description</td>
</tr>
<tr>
<td>get_cost_ip_dc_hrg</td>
</tr>
<tr>
<td>get_doses_combination</td>
</tr>
<tr>
<td>get_doses_combination_units</td>
</tr>
<tr>
<td>get_eqsd_details</td>
</tr>
<tr>
<td>get_extension_file</td>
</tr>
<tr>
<td>get_gender_details</td>
</tr>
<tr>
<td>get_mean_sd_age</td>
</tr>
<tr>
<td>get_mortality_from_file</td>
</tr>
<tr>
<td>get_name_value_probdistb_def</td>
</tr>
<tr>
<td>get_outcome_details</td>
</tr>
<tr>
<td>get_parameter_def_distribution</td>
</tr>
<tr>
<td>get_parameter_direct</td>
</tr>
<tr>
<td>get_parameter_estimated_regression</td>
</tr>
<tr>
<td>get_parameter_read</td>
</tr>
<tr>
<td>get_single_col_multiple_pattern</td>
</tr>
<tr>
<td>get_slope_intercept</td>
</tr>
<tr>
<td>get_slope_intercept_cross</td>
</tr>
<tr>
<td>get_slope_intercept_nested</td>
</tr>
<tr>
<td>get_timepoint_details</td>
</tr>
<tr>
<td>get_trial_arm_details</td>
</tr>
<tr>
<td>get_var_state</td>
</tr>
<tr>
<td>health_state</td>
</tr>
<tr>
<td>init_trace</td>
</tr>
<tr>
<td>init_trace_sjtime</td>
</tr>
<tr>
<td>keep_results_plot_dsa</td>
</tr>
<tr>
<td>list_paramwise_psa_result</td>
</tr>
<tr>
<td>load_trial_data</td>
</tr>
<tr>
<td>map_eqsd5Lto3L_VanHout</td>
</tr>
<tr>
<td>map_resource_use_categories</td>
</tr>
<tr>
<td>markov_model</td>
</tr>
<tr>
<td>markov_model_sojourn_time</td>
</tr>
</tbody>
</table>
R topics documented:

- microcosting_liquids_long ................................................. 92
- microcosting_liquids_wide .................................................. 94
- microcosting_patches_long .................................................. 97
- microcosting_patches_wide .................................................. 99
- microcosting_tablets_long .................................................. 101
- microcosting_tablets_wide .................................................. 103
- plot_ceac_psa ................................................................. 105
- plot_dsa ................................................................. 105
- plot_dsa_difference ......................................................... 107
- plot_dsa_icer_range ......................................................... 108
- plot_dsa_nmb_range ......................................................... 109
- plot_dsa_others_range ....................................................... 109
- plot_efficiency_frontier ..................................................... 110
- plot_histogram_onetimepoint_twogroups .................................. 110
- plot_meanSE_longitudinal_twogroups ..................................... 111
- plot_model ................................................................. 112
- plot_nmb_lambda .............................................................. 113
- plot_prediction_parametric_survival ...................................... 114
- plot_return_residual_cox .................................................... 115
- plot_return_residual_survival .............................................. 116
- plot_return_survival_curve .................................................. 117
- plot_survival_cox_covariates .............................................. 118
- populate_transition_matrix .................................................. 119
- predict_coxph ............................................................... 119
- promis3a_scoring.df .......................................................... 121
- report_sensitivity_analysis .................................................. 121
- return0_if_not_null_na ....................................................... 122
- return_equal_liststring_col ................................................ 123
- return_equal_liststring_listcol .......................................... 124
- return_equal_str_col ........................................................ 125
- set_var_state ............................................................... 125
- strategy ................................................................. 126
- summary_plot_psa ............................................................. 127
- table_param.df .............................................................. 128
- trace_data.df ............................................................... 129
- transition_cost_util ......................................................... 129
- trial_data.df ............................................................... 130
- use_coxph_survival ............................................................ 130
- use_fh2_survival ............................................................... 132
- use_fh_survival ............................................................... 133
- use_generalised_linear_mixed_model ...................................... 134
- use_generalised_linear_model .............................................. 135
- use_km_survival ............................................................... 136
- use_linear_mixed_model ..................................................... 138
- use_linear_regression ......................................................... 139
- use_parametric_survival ..................................................... 140
- use_seemingly_unrelated_regression ..................................... 142
- use_survival_analysis ........................................................ 143
add_entries_sameuse_timepoint

Function to get sum of entries of resource per individual at diff timepoints if same category has listed multiple time for same id of participant, this method comes in handy to get the sum.

Description

Function to get sum of entries of resource per individual at diff timepoints if same category has listed multiple time for same id of participant, this method comes in handy to get the sum.

Usage

add_entries_sameuse_timepoint(
    use_data,
    timepointcol,
    timepointval,
    idcolumn,
    result_col
)

Arguments

use_data the data where the observations are held

The data where the timepoints are noted

timepointval which time point is considered now at which the descriptive analysis is done

idcolumn id for each participant

result_col name of the column where the sum of entries to be saved

Value

the data with added sum of resource use
assign_parameters

Description
Function to assign the values of nested parameters from the parameter list

Usage
assign_parameters(param_list)

Arguments
param_list list of parameters, can be nested, or can be used the list returned from define_parameters()
Details

The parameter list should be a list of parameters in the form name value pairs. If the name value pairs is given as a string it throws error as in assign_parameters(c("cost_A = 100", "a = 10")) even if you use assign_parameters(define_parameters(c("cost_A = 100","a = 10"))) but this will be ok if you use the below forms assign_list2 <- c(a = 10, cost_A = "a + 100", cost_B = 10) assign_parameters(assign_list2) OR param_list <- define_parameters(a = 10, cost_A = "a + 100", cost_B = 10) assign_list <- assign_parameters(param_list) Also for nested parameters, remember to give the parameters in order so that at run time, the parameters can be evaluated for example, assign_list = define_parameters(cost_A="a+100",a=10) assign_parameters(assign_list) will throw an error, while assign_list = define_parameters(a=10,cost_A="a+100") assign_parameters(assign_list) will successfully assign parameters as the parameters 'a' is visible before the calculation of 'cost_A'.

Another thing to note is that while using define_parameters, just enumerate them, no need to create as a list by using c() or list function.

Value

list of assigned parameters

Examples

```r
param_list <- define_parameters(
  cost_direct_med_A = 1701, cost_comm_care_A = 1055,
  cost_direct_med_B = 1774, cost_comm_care_B = 1278,
  cost_direct_med_C = 6948,
  cost_comm_care_C = 2059, cost_zido = 2456, cost_health_A =
  "cost_direct_med_A + cost_comm_care_A",
  cost_health_B = "cost_direct_med_B + cost_comm_care_B",
  cost_health_C = "cost_direct_med_C + cost_comm_care_C",
  cost_drug = "cost_zido"
)
assign_parameters(param_list)
```

---

**blank.df**  
Parameter table created

Description

Parameter table created

Usage

blank.df

Format

A 2 column 1 observation
calculate_icer_nmb  

Estimation of ICER and NMB

Description

Estimation of ICER and NMB

Usage

```
calculate_icer_nmb(list_markov, threshold, comparator = NULL)
```

Arguments

- `list_markov`: list of Markov model objects with their Markov trace, cost matrix and utility matrix
- `threshold`: threshold value of WTP
- `comparator`: the strategy to be compared with

Value

ICER and NMB for all the strategies compared to comparator

Examples

```r
well <- health_state("well", cost = 0, utility = 1)
disabled <- health_state("disabled", cost = 100, utility = 1)
dead <- health_state("dead", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead")
tm <- populate_transition_matrix(3, tmat, c(0.6, 0.2, 0.2, 0.6, 0.4, 1), colnames(tmat))
health_states <- combine_state(well, disabled, dead)
this.strategy <- strategy(tm, health_states, "control")
this_markov <- markov_model(this.strategy, 24, c(1000, 0, 0), c(0,0))
well <- health_state("well", cost = 0, utility = 1)
disabled <- health_state("disabled", cost = 10, utility = 0.5)
dead <- health_state("dead", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead")
tm <- populate_transition_matrix(3, tmat, c(0.4, 0.4, 0.2, 0.6, 0.4, 1), colnames(tmat))
health_states <- combine_state(well, disabled, dead)
this.strategy <- strategy(tm, health_states, "intervention")
sec_markov <- markov_model(this.strategy, 24, c(1000, 0, 0), c(0,0))
```
checks_markov_pick_method

list_markov <- combine_markov(this_markov, sec_markov)
calculate_icer_nmb(list_markov, 20000, comparator = "control")

checks_markov_pick_method

*Checks the input to run the Markov cycles and picks correct method*

**Description**

Checks the input to run the Markov cycles and picks correct method

**Usage**

```r
checks_markov_pick_method(
  current_strategy,
  initial_state,
  discount,
  method,
  half_cycle_correction,
  startup_cost,
  startup_util,
  state_cost_only_prevalent,
  state_util_only_prevalent
)
```

**Arguments**

- `current_strategy`: strategy object
- `initial_state`: value of states initially
- `discount`: rate of discount for costs and qalys
- `method`: what type of half cycle correction needed
- `half_cycle_correction`: boolean to indicate half cycle correction
- `startup_cost`: cost of states initially
- `startup_util`: utility of states initially if any
- `state_cost_only_prevalent`: boolean parameter to indicate if the costs for state occupancy is only for those in the state excluding those that transitioned new. This is relevant when the transition cost is provided for eg. in a state with dialysis the cost of previous dialysis is different from the newly dialysis cases. Then the `state_cost_only_prevalent` should be TRUE
- `state_util_only_prevalent`: boolean parameter to indicate if the utilities for state occupancy is only for those in the state excluding those that transitioned new.
Value

changed method name

Examples

tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
tm <- populate_transition_matrix(2, tmat, c(0.5, 0.5, 0, 1))
a <- health_state("Healthy", 1, 1, 0, FALSE)
b <- health_state("Dead", 1, 0, 0, TRUE)
health_states <- combine_state(a, b)
this.strategy <- strategy(tm, health_states, "intervention")
checks_markov_pick_method(this.strategy, c(1, 0), c(0, 0),
"half cycle correction", TRUE, NULL, NULL)

checks_plot_dsa  Function to do some checks before plotting sensitivity analysis results

Description

Function to do some checks before plotting sensitivity analysis results

Usage

checks_plot_dsa(
  result_dsa_control,
  plotfor,
  type,
  result_dsa_treat,
  threshold,
  comparator
)

Arguments

result_dsa_control  result from deterministic sensitivity analysis for first or control model
plotfor  the variable to plotfor e.g. cost, utility NMB etc
type  type of analysis, range or difference
result_dsa_treat  result from deterministic sensitivity analysis for the comparative Markov model
threshold  threshold value of WTP
comparator  the strategy to be compared with
check_equal_columncontents_NAomitted

Function to check the equality of column contents between two data sets with omitting NA

Description

Function to check the equality of column contents between two data sets with omitting NA

Usage

check_equal_columncontents_NAomitted(
  data1,
  data2,
  samecol,
  column_data1,
  column_data2
)

Arguments

data1 first data set

data2 second data set

samecol a unique col in both datasets, like tno or id

column_data1 column name in data 1 to be compared

column_data2 column name in data 2 to be compared

Value

0 if they are equal else error message

Examples

eg_data <- as.data.frame(list(no = c(1, 2, 3, 4),
  mark_at_1 = c(12, 7, 23, 45), gender = c("M", "F", "M", "F"),
  mark_at_2 = c(12, 34, 89, 45), trialarm = c("1","1","2","2"),
  time = c(1, 1, 2, 2), id = c(1, 1, 1, 2))

eg_data2 <- as.data.frame(list(no = c(1, 2, 3, 4),
  mark_at_1 = c(12, 27, 23, 45), gender = c("M", "F", "M", "F"),
  mark_at_2 = c(12, 34, 89, 45), trialarm = c("1","1","2","2"),
  time = c(1, 1, 2, 2), id = c(1, 1, 1, 2))

check_equal_columncontents_NAomitted(neg_data, neg_data2, "no",
  "mark_at_1","mark_at_2")
check_equal_sumcolumncontents_NAomitted

Function to check the sum of column contents between two data sets with omitting NA

Description

Function to check the sum of column contents between two data sets with omitting NA

Usage

check_equal_sumcolumncontents_NAomitted(
  data1,
  data2,
  samecol,
  column_data1,
  column_data2
)

Arguments

data1       first data set
data2       second data set
samecol     a unique col in both datasets, like tno or id
column_data1 column name in data 1 to be compared
column_data2 column name in data 2 to be compared

Value

0 if they are equal else error message

Examples

eg_data <- as.data.frame(list(no = c(1, 2, 3, 4),
  mark_at_1 = c(12, 7, 23, 45), gender = c("M", "F", "M", "F"),
  mark_at_2 = c(12, 34, 89, 45), trialarm = c("1", "1", "2", "2"),
  time = c(1,1,2,2), id = c(1, 1, 1, 2))
eg_data2 <- as.data.frame(list(no = c(1, 2, 3, 4),
  mark_at_1 = c(12, 27, 23, 45), gender = c("M", "F", "M", "F"),
  mark_at_2 = c(12, 34, 89, 45), trialarm = c("1", "1", "2", "2"),
  time = c(1,1,2,2), id = c(1, 1, 1, 2))
check_equal_sumcolumncontents_NAomitted(eg_data, eg_data2, "no",
  "mark_at_1","mark_at_2")
check_link_glm

*Function to find the keyword for family of distribution in glm*

---

**Description**

Function to find the keyword for family of distribution in glm

**Usage**

`check_link_glm(family, link)`

**Arguments**

- `family` : family of distribution
- `link` : function to be used

**Details**

Check and get the link function for the method glm

**Value**

the link if they can be accepted else error

**Examples**

```r
check_link_glm("gaussian", "identity")
```

---

check_list_markov_models

*check the list of Markov models*

---

**Description**

check the list of Markov models

**Usage**

`check_list_markov_models(list_markov)`

**Arguments**

- `list_markov` : list of Markov model objects with their Markov trace, cost matrix and utility matrix
check_null_na

Function to check the variable null or NA

Description
Function to check the variable null or NA

Usage
check_null_na(variable)

Arguments
variable name of variable or list of variable to check

Value
-1 or -2 as error, else return 0 as success
check_trans_prob

Examples

```r
var = c("a")
check_null_na(var)
```

---

```
check_trans_prob  Check the transition probabilities for numeric values and unity row sum
```

Description

Check the transition probabilities for numeric values and unity row sum

Usage

```r
check_trans_prob(trans_mat)
```

Arguments

- `trans_mat` transition matrix

Details

checking for rowsum - checks for the class of transition matrix, value of rowsum (to be 1) and numeric values

Value

0 if they add to 1 else error

Examples

```r
tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
tm <- populate_transition_matrix(2, tmat, list_prob = c(0.5, 0.5, 0, 1))
check_trans_prob(tm)
```
check_treatment_arm  
*Function to return treatment arm*

**Description**

Function to return treatment arm

**Usage**

```r
check_treatment_arm(arm)
```

**Arguments**

- `arm` the arm of the trial

**Value**

0, if success -1, if failure

**Examples**

```r
check_treatment_arm("control")
```

---

check_values_states  
*Check if the values of health states are provided*

**Description**

Check if the values of health states are provided

**Usage**

```r
check_values_states(health_states)
```

**Arguments**

- `health_states` list of health_state objects

**Details**

This is to check if the values are numeric during the run time, else to throw an error

**Value**

true or false
**Examples**

```r
well <- health_state("well", cost = 0, utility = 1)
disabled <- health_state("disabled", cost = 100, utility = 1)
dead <- health_state("dead", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead")
tm <- populate_transition_matrix(3, tmat, c(0.6, 0.2, 0.2, 0.6, 0.4, 1))
health_states <- combine_state(well, disabled, dead)
check_values_states(health_states)
```

**Description**

Join Markov model objects

**Usage**

```r
combine_markov(markov1, ...)
```

**Arguments**

- `markov1`: object 1 of class markov_model
- `...`: any additional objects

**Details**

Combining Markov models for easiness of comparison

**Value**

joined objects of type markov_model

**Examples**

```r
well <- health_state("well", cost = 0, utility = 1)
disabled <- health_state("disabled", cost = 10, utility = 0.5)
```
dead <- health_state("dead", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead")
tm <- populate_transition_matrix(3, tmat, c(0.4, 0.4, 0.2, 0.6, 0.4, 1))
health_states <- combine_state(well, disabled, dead)
this.strategy <- strategy(tm, health_states, "example")
sec_markov <- markov_model(this.strategy, 24, c(1000, 0, 0))
list_markov <- combine_markov(this.markov, sec_markov)

---

**combine_state**  
*Join health states*

**Description**

Join health states

**Usage**

combine_state(...)  

**Arguments**

... any additional objects

**Details**

checking each state is a health state and join them

**Value**

joined health states

**Examples**

a <- health_state("IT", 100, 0.4, 0, FALSE)
b <- health_state("PT", 100, 0.4, 0, FALSE)
combine_state(a, b)
**convert_freq_diff_basis**

*Convert frequency medication to given basis*

**Description**
Convert frequency medication to given basis

**Usage**

```r
convert_freq_diff_basis(freq_given, basis = "day")
```

**Arguments**

- `freq_given`: given frequency
- `basis`: given basis, default is daily

**Value**
converted frequency

**Examples**

```r
correct_freq_diff_basis("once daily")
correct_freq_diff_basis("bd", "week")
correct_freq_diff_basis("Every 4 days", "day")
```

---

**convert_to_given_timeperiod**

*Convert period to given basis*

**Description**
Convert period to given basis

**Usage**

```r
correct_to_given_timeperiod(given_time, basis_time = "day")
```

**Arguments**

- `given_time`: given time
- `basis_time`: given basis, default is "day"
convert_weight_diff_basis

Value
converted unit

Examples
convert_to_given_timeperiod("4 weeks")
convert_to_given_timeperiod("a month")
convert_to_given_timeperiod("1 week")

convert_volume_basis  Convert volume to given basis

Description
Convert volume to given basis

Usage
convert_volume_basis(given_unit, basis = "ml")

Arguments
given_unit given unit
basis given basis, default is "ml"

Value
converted unit

Examples
convert_volume_basis("ml", "liter")

convert_weight_diff_basis  Convert unit strength to given basis

Description
Convert unit strength to given basis

Usage
convert_weight_diff_basis(given_unit, basis = "mg")
**convert_wtpertimediff_basis**

**Arguments**

- `given_unit`  
  given unit
- `basis`  
  given basis, default is "mg"

**Value**

converted unit

**Examples**

```
convert_weight_diff_basis("mg")
convert_weight_diff_basis("kilogram", "micro gram")
```

**Description**

Convert weight per time to given basis

**Usage**

```
convert_wtpertimediff_basis(given_unit, basis = "mcg/hour")
```

**Arguments**

- `given_unit`  
  given unit
- `basis`  
  given basis, default is "mg"

**Value**

converted unit

**Examples**

```
convert_wtpertimediff_basis("mg/day")
convert_wtpertimediff_basis("mcg/day")
convert_wtpertimediff_basis("mg/hour")
```
convert_wtpervoldiff_basis

*Convert wt per unit volume to given basis*

**Description**

Convert wt per unit volume to given basis

**Usage**

convert_wtpervoldiff_basis(given_unit, basis = "mg/ml")

**Arguments**

- given_unit: given unit
- basis: given basis, default is "mg/ml"

**Value**

converted unit

**Examples**

convert_wtpervoldiff_basis("g/ml")

costing_AandE_admission

*Function to estimate the cost of inpatient admission but taken from GP records where code or description known*

**Description**

Function to estimate the cost of inpatient admission but taken from GP records where code or description known

**Usage**

costing_AandE_admission(
    ind_part_data,
    code_ae,
    descrip_ae,
    number_use_ae,
    type_admit_ae,
    unit_cost_data,
    code_col,
costing_AandE_admission

```r
type_admit_col,
description_col,
unit_cost_col,
cost_calculated_in = "attendance",
sheet = NULL
)
```

**Arguments**

- **ind_part_data** (IPD)
- **code_ae** column name of code (for inpatient admission)
- **descrip_ae** column name of description for inpatient admission
- **number_use_ae** the number of days spent in each admission if that is a criteria to be included. Otherwise each admission will be costed
- **type_admit_ae** term indicating admission and type of attendance
- **unit_cost_data** unit cost data file with code/descriptions and unit costs are listed for inpatient admission
- **code_col** code column name in unit cost data
- **type_admit_col** colname that describes type of the attendance and
- **description_col** column name of description of inpatient admission in the unit cost data
- **unit_cost_col** column name of unit cost in unit_cost_data
- **cost_calculated_in** name of unit where the cost is calculated assumed to be per admission
- **sheet** sheet where the unit costs are listed in the unit costs data file

**Value**

the calculated cost of inpatient admission long with original data

**Examples**

```r
costs_file <- system.file("extdata", "National_schedule_of_NHS_costs_2019_AandE.csv", package = "packDAMipd")
datafile <- system.file("extdata", "resource_use_ae_ip.csv", package = "packDAMipd")
ind_part_data <- packDAMipd::load_trial_data(datafile)
unit_cost_data <- packDAMipd::load_trial_data(costs_file)
result <- costing_AandE_admission(ind_part_data = ind_part_data,
code_ae = "code", descrip_ae = NULL, number_use_ae = "number_use",
type_admit_ae = "type_admit", unit_cost_data = unit_cost_data,
description_col = NULL, unit_cost_col = "National_Average_Unit_Cost",
cost_calculated_in = "attendance")
```
costing_inpatient_daycase_admission

Function to estimate the cost of inpatient admission but taken from GP records where HRG code or description known

Description

Function to estimate the cost of inpatient admission but taken from GP records where HRG code or description known

Usage

costing_inpatient_daycase_admission(
    ind_part_data,
    hrg_code_ip_admi,
    descrip_ip_admi,
    number_use_ip_admi,
    elective_col,
    unit_cost_data,
    hrg_code_col,
    description_col,
    unit_cost_col,
    cost_calculated_in = "admission"
)

Arguments

ind_part_data  IPD
hrg_code_ip_admi  column name of hrg code (for inpatient admission)
descrip_ip_admi  column name of description for inpatient admission
number_use_ip_admi  the number of days spent in each admission if that is a criteria to be included. Otherwise each admission will be costed
elective_col  colname to say whether it is an elective admission or non elective admission
unit_cost_data  unit cost data file with hrg code/descriptions and unit costs are listed for inpatient admission
hrg_code_col  hrg code column name in unit cost data
description_col  column name of description of inpatient admission in the unit cost data
unit_cost_col  column name of unit cost in unit_cost_data
cost_calculated_in  name of unit where the cost is calculated assumed to be per admission
Value

the calculated cost of inpatient admission long with original data

Examples

datafile <- system.file("extdata", "resource_use_hc_ip.csv", package = "packDAMipd")
ind_part_data <- packDAMipd::load_trial_data(datafile)
unit_cost_data <- packDAMipd::load_trial_data(costs_file)
result <- costing_inpatient_daycase_admission(ind_part_data, hrg_code_ip_admi = "HRGcode", descrip_ip_admi = NULL, number_use_ip_admi = "number_use", elective_col = "EL", unit_cost_data, hrg_code_col = "Currency_Code", description_col = NULL, unit_cost_col = "National_Average_Unit_Cost", cost_calculated_in = "admission")

---

costing_opioid_liquids_averageMED_long

Function to estimate the cost of liquids when IPD is in long format

Description

Function to estimate the cost of liquids when IPD is in long format

Usage

costing_opioid_liquids_averageMED_long(
  the_columns,
  ind_part_data_long,
  name_med,
  brand_med = NULL,
  bottle_size,
  bottle_size_unit = NULL,
  bottle_lasts,
  bottle_lasts_unit = NULL,
  preparation_dose,
  preparation_unit = NULL,
  timeperiod,
  unit_cost_data,
  unit_cost_column,
  cost_calculated_per,
  strength_column,
  list_of_code_names = NULL,
  list_of_code_brand = NULL,
  list_of_code_bottle_size_unit = NULL,
Arguments

the_columns      columns that are to be used to convert the data from long to wide
ind_part_data_long  IPD
name_med        name of medication
brand_med       brand name of medication if revealed
bottle_size     size of the bottle used
bottle_size_unit  unit of bottle volume
bottle_lasts    how long the bottle lasted
bottle_lasts_unit   time unit of how long the bottle lasted
preparation_dose dose if preparation is given
preparation_unit  unit of preparation dose
timeperiod      time period for cost calculation
unit_cost_data  unit costs data
unit_cost_column column name of unit cost in unit_cost_data
cost_calculated_per column name of unit where the cost is calculated
strength_column column name that has strength of medication
list_of_code_names if names is coded, give the code:name pairs, optional
list_of_code_brand if brand names are coded, give the code:brand pairs, optional
list_of_code_bottle_size_unit list of bottle size units and codes
list_of_code_bottle_lasts_unit list of time of bottle lasts and codes
list_preparation_dose_unit list of preparation dose units and codes
eqdose_covtab    table to get the conversion factor for equivalent doses, optional
basis_strength_unit strength unit to be taken as basis required for total medication calculations
Value

the calculated cost of tablets along with original data

Examples

```r
med_costs_file <- system.file("extdata", "medication_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication_liq_brand_empty.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx", package = "packDAMipd")
table <- load_trial_data(conv_file)
res <- costing_opioid_liquids_averageMED_wide(
  ind_part_data = ind_part_data, name_med = "liq_name",
  brand_med = "liq_brand", bottle_size = "liq_bottle_size",
  bottle_size_unit = NULL, bottle_lasts = "liq_lasts",
  bottle_lasts_unit = NULL, preparation_dose = "liq_strength",
  preparation_unit = NULL, timeperiod = "1 day",
  unit_cost_data = med_costs, unit_cost_column = "UnitCost",
  cost_calculated_per = "Basis", strength_column = "Strength",
  list_of_code_names = NULL, list_of_code_brand = NULL,
  list_of_code_bottle_size_unit = NULL,
  list_of_code_bottle_lasts_unit = NULL,
  list_preparation_dose_unit = NULL, eqdose_covtab = table,
  basis_strength_unit = NULL)
```

---

costing_opioid_liquids_averageMED_wide

*Function to estimate the cost of liquids taken (from IPD)*

Description

Function to estimate the cost of liquids taken (from IPD)

Usage

```r
costing_opioid_liquids_averageMED_wide(
  ind_part_data,
  name_med,
  brand_med = NULL,
  bottle_size,
  bottle_size_unit = NULL,
  bottle_lasts,
  bottle_lasts_unit = NULL,
  preparation_dose,
  preparation_unit = NULL,
  ...)```
timeperiod,
unit_cost_data,
unit_cost_column,
cost_calculated_per,
strength_column,
list_of_code_names = NULL,
list_of_code_brand = NULL,
list_of_code_bottle_size_unit = NULL,
list_of_code_bottle_lasts_unit = NULL,
list_preparation_dose_unit = NULL,
eqdose_covtab = NULL,
basis_strength_unit = NULL
)

Arguments
ind_part_data   IPD
name_med        name of medication
brand_med       brand name of medication if revealed
bottle_size     size of the bottle used
bottle_size_unit unit of bottle volume
bottle_lasts    how long the bottle lasted
bottle_lasts_unit time unit of how long the bottle lasted
preparation_dose dose if preparation is given
preparation_unit unit of preparation dose
timeperiod      time period for cost calculation
unit_cost_data  unit costs data
unit_cost_column column name of unit cost in unit_cost_data
cost_calculated_per column name of unit where the cost is calculated
strength_column column column name that has strength of medication
list_of_code_names if names is coded, give the code:name pairs, optional
list_of_code_brand if brand names are coded, give the code:brand pairs, optional
list_of_code_bottle_size_unit list of bottle size units and codes
list_of_code_bottle_lasts_unit list of time of bottle lasts and codes
Function to estimate the cost of patches when IPD is in long format using a IPD data of long format

Description

Function to estimate the cost of patches when IPD is in long format using a IPD data of long format
Usage

costing_opioid_patches_averageMED_long(
    the_columns,
    ind_part_data_long,
    name_med,
    brand_med = NULL,
    dose_med,
    unit_med = NULL,
    no_taken,
    freq_taken,
    timeperiod,
    unit_cost_data,
    unit_cost_column,
    cost_calculated_per,
    strength_column,
    list_of_code_names = NULL,
    list_of_code_freq = NULL,
    list_of_code_dose_unit = NULL,
    list_of_code_brand = NULL,
    eqdose_cov_tab = NULL,
    basis_strength_unit = NULL
)

Arguments

the_columns columns that are to be used to convert the data from long to wide
ind_part_data_long IPD
name_med name of medication
brand_med brand name of medication if revealed
dose_med dose of medication used
unit_med unit of medication; use null if its along with the dose
no_taken how many taken
freq_taken frequency of medication
timeperiod time period for cost calculation
unit_cost_data unit costs data
unit_cost_column column name of unit cost in unit_cost_data
cost_calculated_per column name of unit in the cost is calculated
strength_column column column name that contain strength of medication
list_of_code_names if names is coded, give the code:name pairs, optional
costing_opioid_patches_averageMED_wide

list_of_code_freq
   if frequency is coded, give the code:frequency pairs, optional
list_of_code_dose_unit
   if unit is coded, give the code:unit pairs, optional
list_of_code_brand
   if brand names are coded, give the code:brand pairs, optional
eqdose_cov_tab
   table to get the conversion factor for equivalent doses, optional
basis_strength_unit
   strength unit to be taken as basis required for total medication calculations

Value

the calculated cost of tablets along with original data

Examples

med_costs_file <- system.file("extdata", "medication_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx",package = "packDAMipd")
table <- load_trial_data(conv_file)
names <- colnames(ind_part_data)
ending <- length(names)
ind_part_data_long <- tidyr::gather(ind_part_data, measurement, value, names[2]:names[ending], factor_key = TRUE)
the_columns <- c("measurement", "value")
res <- costing_opioid_patches_averageMED_long(the_columns,
   ind_part_data_long = ind_part_data_long, name_med = "patch_name",
   brand_med = "patch_brand", dose_med = "patch_strength",unit_med = NULL,
   no_taken = "patch_no_taken", freq_taken = "patch_frequency",
   timeperiod = "4 months", unit_cost_data = med_costs,
   unit_cost_column = "UnitCost", cost_calculated_per = "Basis",
   strength_column = "Strength", list_of_code_names = NULL,
   list_of_code_freq = NULL, list_of_code_dose_unit = NULL,
   eqdose_cov_tab = table,
   basis_strength_unit = "mcg/hr")

costing_opioid_patches_averageMED_wide

Function to estimate the cost of patches taken (from IPD)

Description

Function to estimate the cost of patches taken (from IPD)
Usage

```r
costing_opioid_patches_averageMED_wide(
  ind_part_data,
  name_med,
  brand_med = NULL,
  dose_med,
  unit_med = NULL,
  no_taken,
  freq_taken,
  timeperiod,
  unit_cost_data,
  unit_cost_column,
  cost_calculated_per,
  strength_column,
  list_of_code_names = NULL,
  list_of_code_freq = NULL,
  list_of_code_dose_unit = NULL,
  list_of_code_brand = NULL,
  eqdose_cov_tab = NULL,
  basis_strength_unit = NULL
)
```

Arguments

- `ind_part_data` - IPD
- `name_med` - name of medication
- `brand_med` - brand name of medication if revealed
- `dose_med` - dose of medication used
- `unit_med` - unit of medication; use null if it's along with the dose
- `no_taken` - how many taken
- `freq_taken` - frequency of medication
- `timeperiod` - time period for cost calculation
- `unit_cost_data` - unit costs data
- `unit_cost_column` - column name of unit cost in `unit_cost_data`
- `cost_calculated_per` - column name of unit where the cost is calculated
- `strength_column` - column name that contains strength of medication
- `list_of_code_names` - if names are coded, give the code:name pairs, optional
- `list_of_code_freq` - if frequency is coded, give the code:frequency pairs, optional
- `list_of_code_dose_unit` - if unit is coded, give the code:unit pairs, optional
Function to estimate the cost of tablets when IPD is in long format

**Description**

Function to estimate the cost of tablets when IPD is in long format
Usage

costing_opioid_tablets_averageMED_long(
    the_columns, 
    ind_part_data_long, 
    name_med, 
    brand_med = NULL, 
    dose_med, 
    unit_med = NULL, 
    no_taken, 
    freq_taken, 
    timeperiod, 
    unit_cost_data, 
    unit_cost_column, 
    cost_calculated_per, 
    strength_column, 
    list_of_code_names = NULL, 
    list_of_code_freq = NULL, 
    list_of_code_dose_unit = NULL, 
    list_of_code_brand = NULL, 
    eqdose_cov_tab = NULL, 
    basis_strength_unit = NULL
)

Arguments

the_columns columns that are to be used to convert the data from long to wide
ind_part_data_long IPD
name_med name of medication
brand_med brand name of medication if revealed
dose_med dose of medication used
unit_med unit of medication; use null if its along with the dose
no_taken how many taken
freq_taken frequency of medication
timeperiod time period for cost calculation
unit_cost_data unit costs data
unit_cost_column column name of unit cost in unit_cost_data
cost_calculated_per column name of unit where the cost is calculated
strength_column column column name that contain strength of medication
list_of_code_names if names is coded, give the code:name pairs, optional
Function to estimate the cost of tablets taken as an average cost per equivalent dose in the opioid scenario is the morphine equivalent dose (from IPD)

Description

Function to estimate the cost of tablets taken as an average cost per equivalent dose in the opioid scenario is the morphine equivalent dose (from IPD)
Usage

costing_opioid_tablets_averageMED_wide(
  ind_part_data,
  name_med,
  brand_med = NULL,
  dose_med,
  unit_med = NULL,
  no_taken,
  freq_taken,
  timeperiod,
  unit_cost_data,
  unit_cost_column,
  cost_calculated_per,
  strength_column,
  list_of_code_names = NULL,
  list_of_code_freq = NULL,
  list_of_code_dose_unit = NULL,
  list_of_code_brand = NULL,
  eqdose_cov_tab = NULL,
  basis_strength_unit = NULL
)

Arguments

ind_part_data  IPD
name_med       name of medication
brand_med      brand name of medication if revealed
dose_med       dose of medication used
unit_med       unit of medication; use null if its along with the dose
no_taken       how many taken
freq_taken     frequency of medication
timeperiod     time period for cost calculation
unit_cost_data unit costs data
unit_cost_column
  column name of unit cost in unit_cost_data
cost_calculated_per
  column name of unit where the cost is calculated
strength_column
  column column name that contain strength of medication
list_of_code_names
  if names is coded, give the code:name pairs, optional
list_of_code_freq
  if frequency is coded, give the code:frequency pairs, optional
list_of_code_dose_unit
  if unit is coded, give the code:unit pairs, optional
costing_opioid_tablets_MED_wide

Function to estimate the cost of tablets taken (from IPD)

Description

Function to estimate the cost of tablets taken (from IPD)
Usage

costing_opioid_tablets_MED_wide(
  ind_part_data,
  name_med,
  brand_med = NULL,
  dose_med,
  unit_med = NULL,
  no_taken,
  freq_taken,
  timeperiod,
  unit_cost_data,
  unit_cost_column,
  cost_calculated_per,
  strength_column,
  list_of_code_names = NULL,
  list_of_code_freq = NULL,
  list_of_code_dose_unit = NULL,
  list_of_code_brand = NULL,
  eqdose_cov_tab = NULL,
  basis_strength_unit = NULL
)

Arguments

  ind_part_data   IPD
  name_med        name of medication
  brand_med       brand name of medication if revealed
  dose_med        dose of medication used
  unit_med        unit of medication; use null if its along with the dose
  no_taken        how many taken
  freq_taken      frequency of medication
  timeperiod      time period for cost calculation
  unit_cost_data  unit costs data
  unit_cost_column column name of unit cost in unit_cost_data
  cost_calculated_per column name of unit where the cost is calculated
  strength_column column column name that contain strength of medication
  list_of_code_names if names is coded, give the code:name pairs, optional
  list_of_code_freq if frequency is coded, give the code:frequency pairs, optional
  list_of_code_dose_unit if unit is coded, give the code:unit pairs, optional
Function to estimate the cost of resource use taken (from IPD)

**Details**

Assumes individual level data has name of medication, dose, dose unit, number taken, frequency taken, and basis time. Assumes unit cost data contains the name of medication, form/type, strength, unit of strength (or the unit in which the cost calculated), preparation, unit cost, size and size unit (in which name, forms, size, size unit, and preparation are not passed on).

**Value**

The calculated cost of tablets along with original data.

**Examples**

```r
med_costs_file <- system.file("extdata", "medication_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication_all_brandNull.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx", package = "packDAMipd")
table <- load_trial_data(conv_file)
res <- costing_opioid_tablets_MED_wide(ind_part_data = ind_part_data,
    name_med = "tab_name", brand_med = "tab_brand", dose_med = "tab_str", 
    unit_med = "tab_unit", no_taken = "tab_no_taken", 
    freq_taken = "tab_frequency", timeperiod = "one day", 
    unit_cost_data = med_costs, unit_cost_column = "UnitCost", 
    cost_calculated_per = "Basis", strength_column = "Strength", 
    list_of_code_names = NULL, list_of_code_freq = NULL, 
    list_of_code_dose_unit = NULL, eqdose_cov_tab = table, 
    basis_strength_unit = "mg")
```

---

**Description**

Function to estimate the cost of resource use taken (from IPD)
Usage

costing_resource_use(
  ind_part_data,
  name_use_col,
  each_length_num_use = NULL,
  each_use_provider_indicator = NULL,
  unit_length_use = "day",
  unit_cost_data,
  name_use_unit_cost,
  unit_cost_column,
  cost_calculated_in,
  list_code_use_indicator = NULL,
  list_code_provider_indicator = NULL
)

Arguments

ind_part_data   IPD
name_use_col    name of the column containing resource use
each_length_num_use
  list of column names that shows length/number of repeated use eg. hospital admission
each_use_provider_indicator
  list of column names that shows the bool indicators for the use of resource if this is to be included for the particular provider, say an nhs hospital use
unit_length_use
  the column name that contains how many or how long used
unit_cost_data
  unit costs data where the assumption is that the unit cost for resources such as hospital use, gp visit are listed in column resource/resource use with unit costs in another column and the units calculated as in another column
name_use_unit_cost
  name of resource use (the column name in the unit cost data is assumed to be name/resource/type etc) in unit cost data
unit_cost_column
  column name of unit cost in unit_cost_data
cost_calculated_in
  column name of unit where the cost is calculated
list_code_use_indicator
  if the column name_use_col shows codes to indicate the resource use provide the list of codes and resource use for eg., list(c("yes", "no", c(1,2)))
list_code_provider_indicator
  column each_use_provider_indicator shows codes to indicate the resource use provide the list of codes and resource use for eg., list(c("yes", "no", c(1,2)))

Value

the calculated cost of resource uses along with original data
cost_data.df

Examples

costs_file <- system.file("extdata", "costs_resource_use.csv",
 package = "packDAMipd")
datafile <- system.file("extdata", "resource_use_hc_2.csv",
 package = "packDAMipd")
ind_part_data <- load_trial_data(datafile)
unit_cost_data <- load_trial_data(costs_file)
res <- costing_resource_use(
    ind_part_data[1, ],
    "hospital_admission_1",
    list("length_1", "length_2"),
    list("nhs_1", "nhs_2"),
    "day",
    unit_cost_data, "Inpatient hospital admissions", "UnitCost",
    "UnitUsed",
    NULL, NULL
)

---

cost_data.df  cost matrix

Description

cost matrix

Usage

cost_data.df

Format

A 11 by 2 dataframe

Source

created on Nov 26, 2019 from
tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <-
c("Healthy", "Dead")
tm <- transition_matrix(2, tmat, c(0.5, 0.5, 0.1, 1))
a <- health_state("Healthy", 1, 1, FALSE)
b <- health_state("Dead", 1, 0, TRUE)
health_states <- combine_state(a, b)
this.strategy <- strategy(tm, health_states, "intervention")

create_new_dataset  
create new dataset while keeping cox regression results and returned coefficients

Description
create new dataset while keeping cox regression results and returned coefficients

Usage
create_new_dataset(var, covar, dataset, categorical)

Arguments
- **var**: variable for which the levels have to be identified usually indep variable
- **covar**: the other covariates
- **dataset**: the dataset where these variables contain
- **categorical**: are these variables categorical? True or false

Value
new data frame

Examples
```r
dataset <- survival::lung
new = create_new_dataset("status", c("age"), dataset, c(FALSE))
```

create_shorttable_from_gtsummary_compare_twogroups_timpoints
Create a table to compare the descriptive analysis (short) from gtsummary of two groups, but at different timepoints

Description
Create a table to compare the descriptive analysis (short) from gtsummary of two groups, but at different timepoints

Usage
```r
create_shorttable_from_gtsummary_compare_twogroups_timpoints(
  variables,
  gtsummary,
  name_use,
  timepoints
)
```
create_table_from_gtsummary_compare_twogroups

Create a table to compare the descriptive analysis from gtsummary of two groups

Description

Create a table to compare the descriptive analysis from gtsummary of two groups

Usage

create_table_from_gtsummary_compare_twogroups(variables, gtsummary, name_use)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables</td>
<td>variables that interested</td>
</tr>
<tr>
<td>gtsummary</td>
<td>a gtsummary object that contains summary parameters</td>
</tr>
<tr>
<td>name_use</td>
<td>name of the variable or category</td>
</tr>
<tr>
<td>timepoints</td>
<td>the timepoints at which the descriptive analysis is done</td>
</tr>
</tbody>
</table>

Value

the table

Examples

eg_data <- as.data.frame(list(no = c(1, 2, 3, 4),
mark_at_1 = c(12, 34, 23, 45), gender = c("M", "F", "M", "F"),
mark_at_2 = c(12, 34, 23, 45)))
outcome_summary <- IPDFileCheck::get_summary_gtsummary(eg_data,
c("gender", "mark_at_1", "mark_at_2"), byvar = "gender")
variables <- "Mark"
k <- create_shorttable_from_gtsummary_compare_twogroups_timpoints(variables,
outcome_summary, "Category", c(1, 2))
define_parameters

Function to return a list of parameters given

Description
Function to return a list of parameters given

Usage
define_parameters(...)

Arguments
...
any parameters set of name value pairs expected

Details
To return a list of parameters For using with assign_parameters() just list or enumerate the parameters, do not use c() or list() to create a data type list

Value
a list of parameters

Examples
define_parameters(rr = 1)
**define_parameters_psa**  Define parameter lists for deterministic sensitivity analysis

**Description**

Define parameter lists for deterministic sensitivity analysis

**Usage**

```
define_parameters_psa(base_param_list, sample_list)
```

**Arguments**

- `base_param_list`  list of parameters that used to define Markov model
- `sample_list`      list of parameter values with their sampling distributions

**Value**

table for probability sensitivity analysis

**Examples**

```
param_list <- define_parameters(
  cost_zido = 2278, cost_direct_med_A = 1701,
  cost_comm_care_A = 1055, cost_direct_med_B = 1774,
  cost_comm_care_B = 1278,
  cost_direct_med_C = 6948, cost_comm_care_C = 2059,
  tpAtoA = 1251 / (1251 + 483),
  tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
  tpAtoD = 17 / (17 + 1717),
  tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
  tpBtoD = 15 / (15 + 1243),
  tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312), tpDtoD = 1,
  cost_health_A = "cost_direct_med_A + cost_comm_care_A",
  cost_health_B = "cost_direct_med_B + cost_comm_care_B",
  cost_health_C = "cost_direct_med_C + cost_comm_care_C",
  cost_drug = "cost_zido"
)
sample_list <- define_parameters(cost_zido = "gamma(mean = 2756,
  sd = sqrt(2756))")
param_table <- define_parameters_psa(param_list, sample_list)
```
Define parameter lists for deterministic sensitivity analysis

Usage

```
define_parameters_sens_anal(param_list, low_values, upp_values)
```

Arguments

- `param_list`: list of parameters that used to define Markov model
- `low_values`: list of lower values of those parameters for whom the sensitivity is to be estimated
- `upp_values`: list of upper values of those parameters for whom the sensitivity is to be estimated

Details

Get the parameter list, min and maximum values of the parameters. The min and max values should have same entries, but they should be contained in `param_list` too. Copy the exact values of parameters that are in `param_list` but not in min and max values

Value

table for sensitivity analysis

Examples

```
param_list <- define_parameters(
  cost_zido = 2278, cost_direct_med_A = 1701,
  cost_comm_care_A = 1055, cost_direct_med_B = 1774,
  cost_comm_care_B = 1278,
  cost_direct_med_C = 6948, cost_comm_care_C = 2059,
  tpAtoA = 1251 / (1251 + 483),
  tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
  tpAtoD = 17 / (17 + 1717),
  tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
  tpBtoD = 15 / (15 + 1243),
  tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312), tpDtoD = 1,
  cost_health_A = "cost_direct_med_A + cost_comm_care_A",
  cost_health_B = "cost_direct_med_B + cost_comm_care_B",
  cost_health_C = "cost_direct_med_C + cost_comm_care_C",
  cost_drug = "cost_zido"
)```
Define the table for transition

**Description**

Define the table for transition

**Usage**

```r
define_transition_table(tmat)
```

**Arguments**

- `tmat` transition matrix in the format as in package `mstate`

**Details**

Generating a table for transition matrix for efficient understanding and checking. The transition matrix in the format as per `mstate` package is transformed to a table. If `tmat` is not a square matrix, it gives error else it spells out the transition number, probability name and from state to state.

**Value**

the transition table with the probabilities

**Examples**

```r
tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
define_transition_table(tmat)
```
do_psa

Function to do probabilistic sensitivity analysis

Description

Function to do probabilistic sensitivity analysis

Usage

do_psa(this_markov, psa_table, num_rep)

Arguments

this_markov  Markov model object
psa_table   table object from define_parameters_psa
num_rep  number of repetitions

Value

result after sensitivity analysis

Examples

param_list <- define_parameters(
  cost_zido = 2278, cost_direct_med_A = 1701,
  cost_comm_care_A = 1055, cost_direct_med_B = 1774,
  cost_comm_care_B = 1278,
  cost_direct_med_C = 6948, cost_comm_care_C = 2059,
  tpAtoA = 1251 / (1251 + 483),
  tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
  tpAtoD = 17 / (17 + 1717),
  tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
  tpBtoD = 15 / (15 + 1243),
  tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312), tpDtoD = 1,
  cost_health_A = "cost_direct_med_A+ cost_comm_care_A",
  cost_health_B = "cost_direct_med_B+ cost_comm_care_B",
  cost_health_C = "cost_direct_med_C+ cost_comm_care_C",
  cost_drug = "cost_zido"
)
A <- health_state("A", cost = "cost_health_A+ cost_drug ", utility = 1)
B <- health_state("B", cost = "cost_health_B + cost_drug", utility = 1)
C <- health_state("C", cost = "cost_health_C + cost_drug", utility = 1)
D <- health_state("D", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3, 4), c(NA, 5, 6, 7), c(NA, NA, 8, 9),
c(NA, NA, NA, 10))
colnames(tmat) <- rownames(tmat) <- c("A", "B", "C", "D")
tm <- populate_transition_matrix(4, tmat, c("tpAtoA", "tpAtoB", "tpAtoC", "tpAtoD"),
do_sensitivity_analysis

Function to do deterministic sensitivity analysis

Description

Function to do deterministic sensitivity analysis

Usage

do_sensitivity_analysis(this_markov, param_table)

Arguments

this_markov  Markov model object
param_table  table object from define_parameters_sens_anal() with parameters (base case value, lower and upper)

Value

result after sensitivity analysis

Examples

```r
param_list <- define_parameters(
cost_zido = 2278, cost_direct_med_A = 1701,
cost_comm_care_A = 1055, cost_direct_med_B = 1774,
cost_comm_care_B = 1278,
cost_direct_med_C = 6948, cost_comm_care_C = 2059,
tpAtoA = 1251 / (1251 + 483),
        tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
        tpAtoD = 17 / (17 + 1717),
        tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
        tpBtoD = 15 / (15 + 1243),
        tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312),
        tpDtoD = 1,
```
cost_health_A = "cost_direct_med_A + cost_comm_care_A",
cost_health_B = "cost_direct_med_B + cost_comm_care_B",
cost_health_C = "cost_direct_med_C + cost_comm_care_C",
cost_drug = "cost_zido")
low_values <- define_parameters(cost_direct_med_B = 177.4,
cost_comm_care_C = 205.9)
upp_values <- define_parameters(cost_direct_med_B = 17740,
cost_comm_care_C = 20590)
A <- health_state("A", cost = "cost_health_A + cost_drug ", utility = 1)
B <- health_state("B", cost = "cost_health_B + cost_drug", utility = 1)
C <- health_state("C", cost = "cost_health_C + cost_drug", utility = 1)
D <- health_state("D", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3, 4), c(NA, 5, 6, 7), c(NA, NA, 8, 9),
c(NA, NA, NA, 10))
rownames(tmat) <- c("A", "B", "C", "D")
tm <- populate_transition_matrix(4, tmat, c("tpAtoA", "tpAtoB", "tpAtoC",
"tpAtoD", "tpBtoB", "tpBtoC", "tpBtoD", "tpCtoC", "tpCtoD", "tpDtoD"),
rownames(tmat))
health_states <- combine_state(A, B, C, D)
mono_strategy <- strategy(tm, health_states, "mono")
mono_markov <- markov_model(mono_strategy, 20, c(1, 0, 0, 0),
discount = c(0.06, 0), param_list)
param_table <- define_parameters_sens_anal(param_list, low_values,
upp_values)
result <- do_sensitivity_analysis(mono_markov, param_table)

---

**encode_codes_data**  
*Function to get the codes and the corresponding entries*

**Description**

Function to get the codes and the corresponding entries

**Usage**

`encode_codes_data(list_code_values, data_column_nos, the_data)`

**Arguments**

- `list_code_values`  
  list of codes and values, given as list of lists

- `data_column_nos`  
  the column numbers of data to look for the entries

- `the_data`  
  the data where to look for

**Value**

weight and vol units
Examples

data_file <- system.file("extdata", "medication_liq_codes.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
data_column_nos = c(2,12)
list_of_code_names = list(c(1, 2),c("Morphine", "Oxycodone"))
encode_codes_data(list_of_code_names, data_column_nos, ind_part_data)

eval_assign_trans_prob

Attribute parameters to probabilities of transition matrix

Description

Attribute parameters to probabilities of transition matrix

Usage

eval_assign_trans_prob(tm, parameter_values)

Arguments

- **tm**: A transition matrix in the format from the package 'mstate'
- **parameter_values**: name value pairs of parameter values in the probability matrix

Details

Once the transition matrix is populated, the probabilities in transition matrix gets evaluated and assigned in this function call. If the entry in transition matrix is NA, replaces it with zero similarly to evaluate and assign health states, the parameter values is expected to be a list from assign_parameter() and define_parameter(). The exception is that if the parameters are defined directly and no nested calculation is required. For eg. assign_list = c(p1 = 0.2, p2 = 0.3, p3 = 0.4, p4 = 0.5) prob <- eval_assign_trans_prob(tmat, assign_list) will work For those with nested calculations, this has to be defined as below assign_list<-assign_parameters(define_parameters(p1 = 0.2, p2 = 0.3, p3 = 0.4, p4 = 0.5)) prob <- eval_assign_trans_prob(tmat, assign_list) The below will give error assign_list <- c(p1=0.1, p2 = "p1 + 0.2", p3=0, p4=0.3) prob <- eval_assign_trans_prob(tmat, assign_list)

Value

the transition table with the probabilities
Examples

```r
tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
tmat <- populate_transition_matrix(2, tmat,
  list_prob = c("p1", "p2", "p3", "p4"))
tmat_assigned <- eval_assign_trans_prob(tmat,
  c(p1 = 0.2, p2 = 0.3, p3 = 0.4, p4 = 0.5))
```

---

**eval_assign_values_states**

*Attribute values in health states*

Description

Attribute values in health states

Usage

```r
eval_assign_values_states(health_states, assigned_param)
```

Arguments

- `health_states` : list of health_state objects
- `assigned_param` : name value pairs of parameter values in the probability matrix expected created using function `assign_parameters()`

Details

Assigning the param is done for the cost and utility if the param is not numeric, check if it can be evaluated at the run time if yes, assign the evaluated numeric value if not get the parameters between operators, and assign the values to each individual parameters and then evaluate. only works for two levels. For the example shown the cost is sum of cost_A and cost_B which will only get added in the call `eval_assign_values_states` While initialising the state "well" it will be only saved as expression `cost_A + cost_B` assigned_param (a list) can be expected to be created using assign_parameters() the exception is if parameter is directly assigned with no nested calculation and no missing parameters. For example assigned_param = c(cost_a = 10, cost_b =10) will be ok but not assigned_param = c(a=10, cost_A = "a+100", cost_B =10) as it requires a nested calculation then use `define_parameters()` with assign_parameters() as in `param_list <- define_parameters(a = 10, cost_A = "a + 100", cost_B = 10) assign_list <- assign_parameters(param_list)`

Value

health states with assigned values
Examples

```r
well <- health_state("well", cost = "cost_A + cost_B", utility = 1)
disabled <- health_state("disabled", cost = 100, utility = 1)
dead <- health_state("dead", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead")
tm <- populate_transition_matrix(3, tmat, c(0.6, 0.2, 0.2, 0.6, 0.4, 1))
health_states <- combine_state(well, disabled, dead)
eval_assign_values_states(health_states, c(cost_A = 100, cost_B = 11))
```

---

**find_glm_distribution**  
*Function to find the keyword for family of distribution in glm*

**Description**

Function to find the keyword for family of distribution in glm

**Usage**

```r
find_glm_distribution(text)
```

**Arguments**

- `text`  
  distribution

**Details**

Find the family for glm method

**Value**

the keyword - the name of distribution

**Examples**

```r
find_glm_distribution("gamma")
```
find_keyword_rand_generation

Function to find the keyword for generating random numbers the distribution

Description

Function to find the keyword for generating random numbers the distribution

Usage

find_keyword_rand_generation(text)

Arguments

text name of the probability distribution

Details

This function returns the keyword for generating random number using a keyword provided that is generally used for prob distribution (but R might require a different keyword)

Value

the keyword that should be used in R for generating random numbers

Examples

find_keyword_rand_generation("gamma")

find_keyword_regression_method

Function to find the keyword for regression methods

Description

Function to find the keyword for regression methods

Usage

find_keyword_regression_method(text, additional_info = NA)

Arguments

text regression method
additional_info additional information required
**find_keyword_regression_method**

This function returns the keyword to use in regression methods. For example, linear regression requires `lm` in R. Some regression methods require additional info and it has to be provided.

**Value**

The keyword that should be used in R for regression analysis.

**Examples**

```r
find_keyword_regression_method("linear")
```

---

**find_parameters_btn_operators**

*Function to return parameters with in a expression containing operators*

**Description**

Function to return parameters with in a expression containing operators.

**Usage**

```r
find_parameters_btn_operators(expr)
```

**Arguments**

- `expr`: an expression

**Details**

This function returns the parameters between the operators if the state value or probabilities are defined as expressions. We need to extract the parameters and then assign. First, the position of all operators are found and then return the parameters separated by those operators. This happens only for one level. `find_parameters_btn_operators("a+b")` provides `a` and `b` but for `find_parameters_btn_operators("mean(a,b)+b")` provides `mean(a,b)` and `b`.

**Value**

Parameters in the expression `expr`.

**Examples**

```r
find_parameters_btn_operators("a+b")
```
Function to find the parameters that determine the probability distribution

Usage
find_required_parameter_combs(name_distri)

Arguments
name_distri name of the probability distribution

Details
For each of the probability distribution we require certain parameters and this function provides that required list of parameters.

Value
the parameters that determine the distribution

Examples
find_required_parameter_combs("gamma")

Function to get sum of multiple columns of observations

Usage
find_rowwise_sum_multiplecol(the_data, colnames, sumcolname)

Arguments
the_data the data where the observations are held
colnames columnname in the data whose sum to be obtained
sumcolname name of the new column where sum to be saved


**Value**

the data with added sum

**Examples**

```r
eg_data <- as.data.frame(list(no = c(1, 2, 3, 4),
mark_at_1 = c(12, 7, 23, 45), gender = c("M", "F", "M", "F"),
mark_at_2 = c(12, 34, 89, 45), trialarm = c("1", "1", "2", "2"),
time = c(1, 1, 2, 2), id = c(1, 1, 1, 2)))
find_rowwise_sum_multiplecol(eg_data, c("mark_at_1","mark_at_2"),
"totalmarks")
```

---

**find_survreg_distribution**

*Function to find the keyword for survreg distribution*

**Description**

Function to find the keyword for survreg distribution

**Usage**

```r
find_survreg_distribution(text)
```

**Arguments**

- `text` distribution

**Details**

For surveg method, find the distribution

**Value**

the keyword - the name of distribution

**Examples**

```r
find_survreg_distribution("weibull")
```
form_expression_glm  

Form expression to use with glm()

Description

Form expression to use with glm()

Usage

form_expression_glm(
  param_to_be_estimated,
  indep_var,
  family,
  covariates,
  interaction,
  naaction,
  link
)

Arguments

param_to_be_estimated  
  parameter of interest

indep_var  
  the independent variable (column name in data file)

family  
  distribution name eg. for logistic regression -binomial

covariates  
  list of covariates

interaction  
  boolean value to indicate interaction in the case of generalised linear models, false by default

naaction  
  action to be taken with the missing values

link  
  link function if not the default for each family

Details

Form expression for the method glm

Value

the formula for glm

Examples

formula <- form_expression_glm("admit",
  indep_var = "gre", family = "binomial",
  covariates = c("gpa", "rank"), interaction = FALSE, naaction = "na.omit",
  link = NA)
**form_expression_lm**  
*Form expression to use with lm()*

**Description**  
Form expression to use with lm()

**Usage**  
form_expression_lm(param_to_be_estimated, indep_var, covariates, interaction)

**Arguments**  
- **param_to_be_estimated**  
  parameter of interest
- **indep_var**  
  the independent variable (column name in data file)
- **covariates**  
  list of covariates
- **interaction**  
  boolean value to indicate interaction in the case of linear regression, false by default

**Details**  
This function helps to create the expression for linear regression model it takes care of covariates and interaction

**Value**  
the formula for lm

**Examples**  

```r
formula <- form_expression_lm("gre", indep_var = "gpa", covariates = NA, interaction = FALSE)
```

---

**form_expression_mixed_model_lme4**  
*Form expression to use with mixed models*

**Description**  
Form expression to use with mixed models
Usage

form_expression_mixed_model_lme4(
  param_to_be_estimated,
  dataset,
  fix_eff,
  fix_eff_interact_vars,
  random_intercept_vars,
  nested_intercept_vars_pairs,
  cross_intercept_vars_pairs,
  uncorrel_slope_intercept_pairs,
  random_slope_intercept_pairs,
  family,
  link
)

Arguments

param_to_be_estimated,
    column name of dependent variable
dataset,        a dataframe
fix_eff,         names of variables as fixed effect predictors
fix_eff_interact_vars,
    if interaction -true
random_intercept_vars,
    names of variables for random intercept
nested_intercept_vars_pairs,
    those of the random intercept variables with nested effect
cross_intercept_vars_pairs,
    those of the random intercept variables with crossed effect
uncorrel_slope_intercept_pairs,
    variables with correlated intercepts
random_slope_intercept_pairs,
    random slopes intercept pairs - this is a list of paired variables
family,         family of distribution for non gaussian distribution of predicted variable
link,           link function for the variance

Details

Form the expression for mixed model

Value

result regression result with plot if success and -1, if failure
**Examples**

datafile <- system.file("extdata", "data_linear_mixed_model.csv", package = "packDAMipd")
dt = utils::read.csv(datafile, header = TRUE)
formula <- form_expression_mixed_model_lme4("extro",
    dataset = dt,
    fix_eff = c("open", "agree", "social"),
    fix_eff_interact_vars = NULL,
    random_intercept_vars = c("school", "class"),
    nested_intercept_vars_pairs = list(c("school", "class")),
    cross_intercept_vars_pairs = NULL,
    uncorrel_slope_intercept_pairs = NULL,
    random_slope_intercept_pairs = NULL, family = "binomial", link = NA)

---

**generate_wt_time_units**

*Function to get the weight and time units*

**Description**

Function to get the weight and time units

**Usage**

generate_wt_time_units()

**Value**

weight and time units

**Examples**

ans <- generate_wt_time_units()

---

**generate_wt_vol_units**

*Function to get the weight and volume units*

**Description**

Function to get the weight and volume units

**Usage**

generate_wt_vol_units()
**get_age_details**

*Function to get the details of the age column*

**Value**

weight and vol units

**Examples**

```r
generate_wt_vol_units()
```

---

**Description**

Function to get the details of the age column

**Usage**

```r
get_age_details(trialdata)
```

**Arguments**

- `trialdata`, data containing individual level trial data

**Details**

expecting the data contains the information on age preferably column names "age", "dob" or "yob" or "date of birth", "year of birth", "birth year" If multiple column names match these, then first match will be chosen.

**Value**

the name of the variable related to age and the unique contents if success, else error

**Examples**

```r
get_age_details(data.frame("Age" = c(21, 15),
"arm" = c("control", "intervention")))
```
get_colnames_codedvalues

Function to keep the column name, coded values and non response code into a dataframe

Description

Function to keep the column name, coded values and non response code into a dataframe

Usage

get_colnames_codedvalues(variable, name, code, nrcode = NA)

Arguments

variable, name of the variable in the column
name, column name
code, coded values
nrcode, code for non response

Value

data frame with all the above information

Examples

get_colnames_codedvalues("arm", "pat_trial_arm", c("Y", "N"))

get_cost_AandE_code

Function to extract the unit hospital inpatient admission by matching code

Description

Function to extract the unit hospital inpatient admission by matching code

Usage

get_cost_AandE_code(
  code,
  type_admit,
  ref_cost_data_file,
  col_name_code,
  unit_cost_col,
  type_admit_col,
  sheet = NULL
)
get_cost_AandE_description

Arguments

- **code**: code for AE attendance
- **type_admit**: term indicating admission and type of attendance
- **ref_cost_data_file**: file that has unit cost
- **col_name_code**: name of the column that has the code
- **unit_cost_col**: name of the column with the unit cost
- **type_admit_col**: colname that describes type of the attendance and that indicates admitted or not
- **sheet**: sheet if excel file is given

Value

- **value**: unit cost the unit cost matching the code

Examples

```r
ref_cost_data_file <- system.file("extdata", "National_schedule_of_NHS_costs_2019_AandE.csv", package = "packDAMipd")
re = get_cost_AandE_code("VB02Z", "T01A", ref_cost_data_file, "Currency_Code","National_Average_Unit_Cost", "Service_Code")
```

get_cost_AandE_description

*Function to extract the unit cost by description of AandE att matching description*

Description

Function to extract the unit cost by description of AandE att matching description

Usage

```r
get_cost_AandE_description(
  description,
  type_admit,
  ref_cost_data_file,
  col_name_description,
  unit_cost_col,
  type_admit_col,
  sheet = NULL
)
```
get_cost_ip_dc_description

Arguments

- **description**  
  description of the AE attendance
- **type_admit**  
  term indicating admission and type of attendance
- **ref_cost_data_file**  
  file that has unit cost
- **col_name_description**  
  name of the column that has the description
- **unit_cost_col**  
  name of the column with the unit cost
- **type_admit_col**  
  colname that describes type of the attendance and
- **sheet**  
  sheet if excel file is given

Value

unit cost the unit cost matching the hrg code

Examples

```r
ref_cost_data_file <- system.file("extdata",
"National_schedule_of_NHS_costs_2019_AandE.csv", package = "packDAMipd")
re = get_cost_AandE_description("Emergency Medicine", "T01A",
ref_cost_data_file, "Currency_Description", "National_Average_Unit_Cost",
"Service_Code")
```

get_cost_ip_dc_description

Function to extract the unit hospital inpatient admission by matching description

Description

Function to extract the unit hospital inpatient admission by matching description

Usage

```r
get_cost_ip_dc_description(  
  description,  
  ref_cost_data_file,  
  col_name_description,  
  unit_cost_col,  
  sheet = NULL  
)
```
get_cost_ip_dc_hrg

Function to extract the unit hospital inpatient admission by matching HRG code

Arguments

description description corresponding to the inpatient admission
ref_cost_data_file file that has unit cost
col_name_description name of the column that has the description
unit_cost_col name of the column with the unit cost
sheet sheet if excel file is given

Value

unit cost the unit cost matching the hrg code

Examples

result <- get_cost_ip_dc_description("Cerebrovascular Accident", ref_cost_data_file, "Currency_Description", "National_Average_Unit_Cost")

get_cost_ip_dc_hrg(hrg, ref_cost_data_file, col_name_hrg_code, unit_cost_col, sheet = NULL)
**get_doses_combination**

**Value**

unit cost the unit cost matching the hrq code

**Examples**

```r
get_cost_ip_dc_hrg("AA22C", ref_cost_data_file, "Currency_Code", "National_Average_Unit_Cost")
```

**get_doses_combination**  
*Convert the combined dose to its individual component numerical value or can be unit/unit*

**Description**

Convert the combined dose to its individual component numerical value or can be unit/unit

**Usage**

```r
get_doses_combination(the_string, separator = "/")
```

**Arguments**

- `the_string`: given combined unit
- `separator`: given character for separation, default is "/"

**Value**

separated texts

**Examples**

```r
get_doses_combination("g/ml")
```
get_doses_combination_units

*Convert the combined dose to its individual component numerical value and units*

### Description

Convert the combined dose to its individual component numerical value and units

### Usage

```r
get_doses_combination_units(the_string, separator = "/")
```

### Arguments

- **the_string**: given combined unit
- **separator**: given character for separation, default is "/"

### Value

separated numerical value and its units

### Examples

```r
get_doses_combination_units("10g/2ml")
```

get_eq5d_details

*Function to get the details of the EQ5D column*

### Description

Function to get the details of the EQ5D column

### Usage

```r
get_eq5d_details(trialdata)
```

### Arguments

- **trialdata**: data containing individual level trial data

### Details

Specific to the EQ5D data - the column names are given as certain sets, Tried to give 15 sets as the column names
**Value**

the name of the variable related to EQ5D and the unique contents if success, else error

**Examples**

```r
get_eq5d_details(data.frame(
  "MO" = c(1, 2), "SC" = c(1, 2), "UA" = c(1, 2),
  "PD" = c(1, 2), "AD" = c(1, 2)
))
```

---

**get_extension_file**  
*Function to get extension of a file name*

**Description**

Function to get extension of a file name

**Usage**

```r
get_extension_file(filename)
```

**Arguments**

- `filename` : name of a file

**Details**

if there is no "." character returns error else returns last characters those after string split using ".:"

**Value**

the extension

**Examples**

```r
get_extension_file("data.txt")
```
get_gender_details  
Function to get the details of the gender column

Description
Function to get the details of the gender column

Usage
get_gender_details(trialdata)

Arguments
trialdata, data containing individual level trial data

Details
expecting the data contains the information on gender preferably column names "gender", "sex" or "male" or "female". If multiple column names match these, then first match will be chosen.

Value
the name of the variable related to gender and the unique contents if success, else error

Examples
get_gender_details(data.frame("Age" = c(21, 15), "sex" = c("m", "f")))

get_mean_sd_age  
Function to return mean age from a data frame

Description
Function to return mean age from a data frame

Usage
get_mean_sd_age(this_data, age_nrcode)

Arguments
this_data the data containing column with age
age_nrcode non response code

Details
Age data is complete with the nr code given and get the mean and sd
get_mortality_from_file

Value
mean and sd, if success -1, if failure

Examples
this_data <- as.data.frame(cbind(num = c(1, 2, 3, 4),
age = c(14, 25, 26, 30)))
get_mean_sd_age(this_data, NA)

get_mortality_from_file
Get the mortality rate values from reading a file

Description
Get the mortality rate values from reading a file

Usage
get_mortality_from_file(paramfile, age, mortality_colname, gender = NULL)

Arguments
paramfile parameter file to get the mortality eg.national life table data
age age to get the age specific data
mortality_colname column name with the mortality rates if it is not gender specific
gender gender details to get the gender specific mortality data

Details
Provides the mortality rates as age and gender dependent Assumes the data contains mortality rate for single year and once it extracted per gender will retrieve single value Age column can consists of range of values, or a particular value also assumes that the mortality rate for each gender is listed under the gender column for gender specific values. if the mortality is not gender specific, the column name should be passed on to the function if gender is not null, mortality_name will be ignored

Value
the paramvalue

Examples
paramfile <- system.file("extdata", "LifeTable_USA_Mx_2015.csv",
package = "packDAMipd"
)
a <- get_mortality_from_file(paramfile, age = 10, mortality_colname =
"total", gender = NULL)
get_name_value_probdistrb_def

Function to return the two parameters from a given expression separated by comma.

Description

Function to return the two parameters from a given expression separated by comma,

Usage

get_name_value_probdistrb_def(expr)

Arguments

expr an expression

Details

It will return the parameters of the distribution separated by commas and given in usual notation as brackets. It will identify those in between first occurrence of "( "and last occurrence of ")" and from the characters in between search for comma to indicate different parameters then it will extract (from those extracted parameters separated by commas) that on the left side of "equal" sign get_name_value_probdistrb_def("gamma(mean = sqrt(2), b =17)") will be ok but get_name_value_probdistrb_def("gamma(shape, scale") and get_name_value_probdistrb_def("gamma(shape =1 & scale =1") will show error

Value

parameters in the expression expr

Examples

get_name_value_probdistrb_def("gamma(mean = 10, sd =1")

get_outcome_details Function to get the details of the outcome column

Description

Function to get the details of the outcome column

Usage

get_outcome_details(trialdata, name, related_words, multiple = FALSE)
get_parameter_def_distribution

Arguments

trialdata, data containing individual level trial data
name, name of the variable
related_words, probable column names
multiple, indicates true if there are multiple columns

Details

if the words related to outcome is given, the function will get the columns and the codes used for
the outcome, the difference here is that certain outcomes can be distributed in multiple columns

Value

the name of the variable related to health outcome (any) and the unique contents if success, else
error

Examples

goodafternoon(1, 2, 3)

get_outcome_details(find outcomes, blacklist, "eq5d", TRUE

get_parameter_def_distribution

Get the definition of given parameter distribution defined in a file

Description

Get the definition of given parameter distribution defined in a file

Usage

goodafternoon(1, 2, 3)

get_parameter_def_distribution(
parameter,
paramfile,
colnames_paramdistr,
strategycol = NA,
strategyname = NA
)
Arguments

- **parameter**: parameter of interest
- **paramfile**: data file to be provided
- **colnames_paramdistr**: list of column names for the parameters that define the distribution
- **strategycol**: treatment strategy column name
- **strategynname**: treatment strategy name in the column strategycol

Details

This function reads the parameter distribution from a file and returns the parameter obtained. It assumes that the file contains parameter, distribution column names for parameter values for the distribution are passed on to the function. Assumes the name of each parameter and value are given in the consecutive columns. Once the expression is created using the parameters given in the file, it gets checked for correctness of specifying the distribution in R context using the function `check_estimate_substitute_proper_params` and then evaluated.

Value

the definition of parameter from the given distribution

Examples

```r
paramfile <- system.file("extdata", "table_param.csv", 
package = "packDAMipd")
a <- get_parameter_def_distribution("rr", paramfile, c("Param1_name",  
"Param1_value"))
```

---

get_parameter_direct  Get the parameter values from reading a file

Description

Get the parameter values from reading a file

Usage

```r
get_parameter_direct(parameter, paramvalue)
```

Arguments

- **parameter**: parameter of interest
- **paramvalue**: parameter value to be assigned

Details

Basic function to assign a parameter directly
get_parameter_estimated_regression

Value

the paramvalue

Examples

a <- get_parameter_direct("cost_IT", paramvalue = 100)

get_parameter_estimated_regression

Get the parameter values using the provided statistical regression methods

Description

Get the parameter values using the provided statistical regression methods

Usage

get_parameter_estimated_regression(
  param_to_be_estimated, 
  data, 
  method, 
  indep_var, 
  info_get_method = NA, 
  info_distribution = NA, 
  covariates = NA, 
  timevar_survival = NA, 
  interaction = FALSE, 
  fix_eff = NA, 
  fix_eff_interact_vars = NA, 
  random_intercept_vars = NA, 
  nested_intercept_vars_pairs = NA, 
  cross_intercept_vars_pairs = NA, 
  uncorrel_slope_intercept_pairs = NA, 
  random_slope_intercept_pairs = NA, 
  naaction = "stats::na.omit", 
  param2_to_be_estimated = NA, 
  covariates2 = NA, 
  interaction2 = FALSE, 
  link = NA, 
  cluster_var = NA, 
  package_mixed_model = NA
)
**Arguments**

- **param_to_be_estimated**
  - parameter of interest
- **data**
  - data to be provided or the data file containing dataset
- **method**
  - method of estimation (for example, linear, logistic regression etc)
- **indep_var**
  - the independent variable (column name in data file)
- **info_get_method**
  - additional information on methods e.g. Kaplan-Meier or hazard
- **info_distribution**
  - distribution name eg. for logistic regression - binomial
- **covariates**
  - list of covariates - calculations to be done before passing
- **timevar_survival**
  - time variable for survival analysis
- **interaction**
  - boolean value to indicate interaction in the case of linear regression
- **fix_eff**
  - boolean value to indicate interaction in the case of linear regression
- **fix_eff_interact_vars**
  - boolean value to indicate interaction in the case of linear regression
- **random_intercept_vars**
  - boolean value to indicate interaction in the case of linear regression
- **nested_intercept_vars_pairs**
  - boolean value to indicate interaction in the case of linear regression
- **cross_intercept_vars_pairs**
  - boolean value to indicate interaction in the case of linear regression
- **uncorrel_slope_intercept_pairs**
  - boolean value to indicate interaction in the case of linear regression
- **random_slope_intercept_pairs**
  - boolean value to indicate interaction in the case of linear regression
- **naaction**
  - what action to be taken for the missing values, default is a missing value.
- **param2_to_be_estimated**
  - parameter of interest for equation 2 in bivariate regression
- **covariates2**
  - list of covariates - for equation 2 in bivariate regression
- **interaction2**
  - boolean value to indicate interaction for equation 2 in bivariate regression
- **link**
  - link function to be provided if not using the default link for each of the info_distribution
- **cluster_var**
  - cluster variable if any
- **package_mixed_model**
  - package to be used for mixed model ie nlme or lme4

**Details**

This function is the top in the layer of functions used for regression analysis. Thus it contains many parameters to be passed on. The required ones are parameter to be estimated, data that contains the observation, the method of regression to be used, the independent variable and the information for the distribution and method. If the data is given as a file name, it will load the data in that file. Then it calls the appropriate functions depending on the regression method that specified. The methods that are considered: Survival analysis, linear regression, logistic regression, generalised linear model, linear multilevel or mixed model, and seemingly unrelated regression.
Value

results the results of the regression analysis

Examples

```r
result <- get_parameter_estimated_regression(
  param_to_be_estimated = "Direction",
  data = ISLR::Smarket, method = "logistic", indep_var = "Lag1",
  info_get_method = NA, info_distribution = "binomial",
  covariates = c("Lag2", "Lag3"), interaction = FALSE,
  naaction = "na.omit", link = NA)
```

---

get_parameter_read  Get the parameter values from reading a file

Description

Get the parameter values from reading a file

Usage

```r
get_parameter_read(parameter, paramfile, strategycol = NA, strategyname = NA)
```

Arguments

- `parameter` parameter of interest
- `paramfile` parameter file to be provided
- `strategycol` treatment strategy
- `strategyname` treatment strategy name in the column strategycol

Details

This function read the parameter from a file given that the file has these column names (at least)
Parameter and Value Strategy col and name are optional. Check if the data file contains column
names parameter and value and then get the results.

Value

- the paramvalue

Examples

```r
a <- get_parameter_read("cost_IT", paramfile = system.file("extdata",
  "table_param.csv", package = "packDAMipd"))
```
get_single_col_multiple_pattern

Function to get cols for the pattern given

Description

Function to get cols for the pattern given

Usage

get_single_col_multiple_pattern(pattern, the_data)

Arguments

pattern the pattern to look for
the_data data where to look at

Value

zero or -1

Examples

the_data <- as.data.frame(cbind(c("one", "two"), c("a", "b"), c("aa", "bb")))
colnames(the_data) <- c("name", "brand_one", "two")
get_single_col_multiple_pattern(c("brand", "trade"), the_data)

get_slope_intercept help function to keep slope and intercept portion ready in mixed model expression

Description

help function to keep slope and intercept portion ready in mixed model expression

Usage

get_slope_intercept(
    expression,
    random_intercept_vars,
    random_slope_intercept_pairs,
    uncorrel_slope_intercept_pairs
)
get_slope_intercept_cross

**Arguments**

- `expression`: expression created so far
- `random_intercept_vars`,
  names of variables for random intercept
- `random_slope_intercept_pairs`,
  random slopes intercept pairs this is a list of paired variables
- `uncorrel_slope_intercept_pairs`,
  variables with correlated intercepts

**Value**

expression expression created

```
get_slope_intercept_cross

**help function to keep slope and intercept portion ready in mixed model expression**
```

**Description**

help function to keep slope and intercept portion ready in mixed model expression

**Usage**

```
get_slope_intercept_cross(
  expression,
  random_intercept_vars,
  intercept_vars_pairs,
  random_slope_intercept_pairs,
  uncorrel_slope_intercept_pairs
)
```

**Arguments**

- `expression`: expression created so far
- `random_intercept_vars`,
  names of variables for random intercept
- `intercept_vars_pairs`,
  those of the random intercept variables with nested effect
- `random_slope_intercept_pairs`,
  random slopes intercept pairs this is a list of paired variables
- `uncorrel_slope_intercept_pairs`,
  variables with correlated intercepts

**Value**

expression expression created
get_slope_intercept_nested

Help function to keep slope and intercept portion ready in mixed model expression

Description

Help function to keep slope and intercept portion ready in mixed model expression

Usage

get_slope_intercept_nested(
  expression,
  random_intercept_vars,
  intercept_vars_pairs,
  random_slope_intercept_pairs,
  uncorrel_slope_intercept_pairs
)

Arguments

text
  expression created so far
random_intercept_vars,
  names of variables for random intercept
intercept_vars_pairs,
  those of the random intercept variables with nested effect
random_slope_intercept_pairs,
  random slopes intercept pairs this is a list of paired variables
uncorrel_slope_intercept_pairs,
  variables with correlated intercepts

Value

expression expression created

get_timepoint_details

Function to get the details of the time point column

Description

Function to get the details of the time point column

Usage

get_timepoint_details(trialdata)
Arguments

trialdata, data containing individual level trial data

Details

expecting the data contains the information on timepoints preferably column names "time point", "times" or "time" or "timepoint". If multiple column names match these, then first match will be chosen.

Value

the name of the variable related to time point and the unique contents if success, else error

Examples

get_timepoint_details(data.frame("time" = c(21, 15), "arm" = c("control", "intervention")))

get_trial_arm_details Function to get the details of the trial arm

Description

Function to get the details of the trial arm

Usage

get_trial_arm_details(trialdata)

Arguments

trialdata, data containing individual level trial data

Details

expecting the data contains the information on trial arm preferably column names "arm", "trial" or "trial arm". If multiple column names match these, then first match will be chosen.

Value

the name of the variable related to trial arm and the unique contents if success, else error

Examples

get_trial_arm_details(data.frame("Age" = c(21, 15), "arm" = c("control", "intervention")))
**get_var_state**  
*Get the attribute for the health state*

**Description**
Get the attribute for the health state

**Usage**
get_var_state(state, var)

**Arguments**
- state: object of class health state
- var: attribute of the health state

**Details**
After checking the given state is a health state and given variable is defined in the health state, the value of the variable is returned

**Value**
modified health state

**Examples**
get_var_state(health_state("IT", 100, 0.4, 0, FALSE), "cost")

---

**health_state**  
*Definition of health state class or health state constructor*

**Description**
Definition of health state class or health state constructor

**Usage**
health_state(name, cost, utility, state_time = 0, absorb = FALSE)

**Arguments**
- name: name of the health state
- cost: value or expression that represents cost of the health state
- utility: value or expression that represents utility of the health state
- state_time: time denoting how long in the state
- absorb: boolean indicating health state absorbing or not
Details

Initialising the name, cost, utility and time spent for the health state name is the name of the health state cost/utility can be defined as characters e.g. "cost_A" if they are characters, the value is assigned after parsing the text. state_time is integer and absorb is boolean

Value

value of the state

Examples

```r
st <- health_state("IT", 100, 0.4, 0, FALSE)
st <- health_state("IT", "cost_A", 0.4, 0, FALSE)
```

---

### init_trace

**Define an all zero trace matrix**

**Description**

Define an all zero trace matrix

**Usage**

```r
init_trace(health_states, cycles)
```

**Arguments**

- `health_states`: health states
- `cycles`: no of cycles

**Details**

Initialise the trace matrix with all zeros trace matrix will be with no_cycles+1 by no_states matrix

**Value**

trace matrix -all zero

**Examples**

```r
a <- health_state("Healthy", 1, 1, 0, FALSE)
b <- health_state("Dead", 1, 0.5, 0, FALSE)
health_states <- combine_state(a, b)
init_trace(health_states, 10)
```
init_trace_sjtime  

*Define an all zero trace matrix*

**Description**
Define an all zero trace matrix

**Usage**

```r
define_trace_sjtime(health_states, cycles)
```

**Arguments**

- `health_states`: health states
- `cycles`: no of cycles

**Details**
Initialise the trace matrix with all zeros. The trace matrix will be of size no_cycles+1 by no_states.

**Value**

- trace matrix - all zero

**Examples**

```r
a <- health_state("Healthy", 1, 1, 0, FALSE)
b <- health_state("Dead", 1, 0.5, 0, FALSE)
health_states <- combine_state(a, b)
init_trace(health_states, 10)
```

---

`keep_results_plot_dsa`  

*Function to do some checks before plotting sensitivity analysis results*

**Description**
Function to do some checks before plotting sensitivity analysis results

**Usage**

```r
keep_results_plot_dsa(
  result_dsa_control,
  plotfor,
  result_dsa_treat,
  plot_variable,
  threshold,
  comparator
)
```
Arguments

- **result_dsa_control**: result from deterministic sensitivity analysis for first or control model
- **plotfor**: the variable to plot for e.g. cost, utility NMB etc
- **result_dsa_treat**: result from deterministic sensitivity analysis for the comparative Markov model
- **plot_variable**: variable for plotting
- **threshold**: threshold value of WTP
- **comparator**: the strategy to be compared with

Value

results to plot dsa

---

**list_paramwise_psa_result**

*Function to list probabilistic sensitivity analysis results parameterwise*

Description

Function to list probabilistic sensitivity analysis results parameterwise

Usage

```r
list_paramwise_psa_result(
  result_psa_params_control,
  result_psa_params_treat,
  threshold,
  comparator
)
```

Arguments

- **result_psa_params_control**: result from probabilistic sensitivity analysis for first or control model
- **result_psa_params_treat**: result from probabilistic sensitivity analysis for the comparative Markov model
- **threshold**: threshold value of WTP
- **comparator**: the strategy to be compared with

Value

plot of sensitivity analysis
Examples

```r
param_list <- define_parameters(
  cost_zido = 2278, cost_direct_med_A = 1701,
  cost_comm_care_A = 1055, cost_direct_med_B = 1774,
  cost_comm_care_B = 1278, cost_direct_med_C = 6948, cost_comm_care_C = 2059,
  tpAtoA = 1251 / (1251 + 483),
  tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
  tpAtoD = 17 / (17 + 1717),
  tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
  tpBtoD = 15 / (15 + 1243),
  tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312), tpDtoD = 1,
  cost_health_A = "cost_direct_med_A + cost_comm_care_A",
  cost_health_B = "cost_direct_med_B + cost_comm_care_B",
  cost_health_C = "cost_direct_med_C + cost_comm_care_C",
  cost_drug = "cost_zido"
)

A <- health_state("A", cost = "cost_health_A + cost_drug ", utility = 1)
B <- health_state("B", cost = "cost_health_B + cost_drug", utility = 1)
C <- health_state("C", cost = "cost_health_C + cost_drug", utility = 1)
D <- health_state("D", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3, 4), c(NA, 5, 6, 7), c(NA, NA, 8, 9),
  c(NA, NA, NA, 10))
colnames(tmat) <- rownames(tmat) <- c("A", "B", "C", "D")
tm <- populate_transition_matrix(4, tmat, c(
  "tpAtoA", "tpAtoB", "tpAtoC", "tpAtoD",
  "tpBtoB", "tpBtoC", "tpBtoD", "tpCtoC", "tpCtoD", "tpDtoD"
), colnames(tmat))
health_states <- combine_state(A, B, C, D)
mono_strategy <- strategy(tm, health_states, "mono")
mono_markov <- markov_model(mono_strategy, 20, initial_state = c(1,0,0,0),
  discount = c(0.06, 0),param_list)
sample_list <- define_parameters(cost_zido = "gamma(mean = 2756,
  sd = sqrt(2756))")
param_table <- define_parameters_psa(param_list, sample_list)
result <- do_psa(mono_markov, param_table, 10)
list_paramwise_psa_result(result, NULL, NULL, NULL)
```

load_trial_data

Function to load the file containing trial data and return it

Description

Function to load the file containing trial data and return it

Usage

```r
load_trial_data(file = NULL, sheet = NULL)
```
map_eq5d5Lto3L_VanHout

Arguments

- file, name of the file in full
- sheet, name of the sheet if excel work book is given

Value

trial data if success, else -1

Examples

```r
load_trial_data(system.file("extdata", "trial_data.csv",
package = "packDAMipd")
)
```

map_eq5d5Lto3L_VanHout

Function to map EQ5D5L scores to EQ5D3L scores and then add to IPD data

Description

Function to map EQ5D5L scores to EQ5D3L scores and then add to IPD data

Usage

```r
map_eq5d5Lto3L_VanHout(ind_part_data, eq5d_nrcode)
```

Arguments

- `ind_part_data` a data frame
- `eq5d_nrcode` non response code for EQ5D5L, default is NA

Value

qaly included modified data, if success -1, if failure

Source

http://eprints.whiterose.ac.uk/121473/1/Devlin_et_al-2017-Health_Economics.pdf

Examples

```r
library(valueEQ5D)
datafile <- system.file("extdata", "trial_data.csv",
package = "packDAMipd")
trial_data <- load_trial_data(datafile)
map_eq5d5Lto3L_VanHout(trial_data, NA)
```
map_resource_use_categories

Function to to read the text form of resource use and replace it with standard texts of resource use ie. some one can describe GP visit as GP surgery visit, surgery visit or general practioners visit etc. Here all these texts should be given in a excel or csv file and then corresponding standard form will be read from the file and will be replaced.

Description

Function to to read the text form of resource use and replace it with standard texts of resource use ie. some one can describe GP visit as GP surgery visit, surgery visit or general practioners visit etc. Here all these texts should be given in a excel or csv file and then corresponding standard form will be read from the file and will be replaced.

Usage

map_resource_use_categories(
    the_data,  
    service_actual,  
    new_column,  
    mapped_data,  
    mapped_use,  
    analysis,  
    replace_only,  
    relevant_column = NULL,  
    check_value_relevant = NULL,  
    nhs_use_column = NULL,  
    check_value_nhs_use = NULL
)

Arguments

the_data      the data where the observations are held
service_actual column name of the actual service use
new_column    the name of the column where the mapped resource use to be
mapped_data   data where the service name and mapped service name has been stored
mapped_use    column name of mapped resource use in mapped_data
analysis      base case or secondary
replace_only  if we want to replace only certain resource use
relevant_column the name of the column where the mapped resource use is indicated as relevant or not
check_value_relevant how is the mapped resource is indicated as relevant by a value
markov_model

nhs_use_column  the name of the column where the mapped resource use comes under NHS or not
check_value_nhs_use
    value that is used to indicated the nhs use

Value
the data with added sum

markov_model  

Definition of Markov model and trace

Description
Definition of Markov model and trace

Usage
markov_model(  
    current_strategy,  
    cycles,  
    initial_state,  
    discount = c(0, 0),  
    parameter_values = NULL,  
    half_cycle_correction = TRUE,  
    state_cost_only_prevalent = FALSE,  
    state_util_only_prevalent = FALSE,  
    method = "half cycle correction",  
    startup_cost = NULL,  
    startup_util = NULL  
)

Arguments

    current_strategy  
        strategy object
    cycles  
        no of cycles
    initial_state  
        value of states initially
    discount  
        rate of discount for costs and qalys
    parameter_values  
        parameters for assigning health states and probabilities
    half_cycle_correction  
        boolean to indicate half cycle correction
state_cost_only_prevalent

boolean parameter to indicate if the costs for state occupancy is only for those in the state excluding those that transitioned new. This is relevant when the transition cost is provided for eg. in a state with dialysis the cost of previous dialysis is different from the newly dialysis cases. Then the state_cost_only_prevalent should be TRUE

state_util_only_prevalent

boolean parameter to indicate if the utilities for state occupancy is only for those in the state excluding those that transitioned new.

method

what type of half cycle correction needed

startup_cost
cost of states initially

startup_util
utility of states initially if any

Details

Use the strategy, cycles, initial state values creating the markov model and trace. As many probabilities /cost/utility value depend on age/time the evaluation and assignment happens during each cycle. At the heart it does a matrix multiplication using the previous row of the trace matrix and the columns of the transition matrix. Also checks for population loss, calculates cumulative costs and qalys (accounts for discounting and half cycle correction)

Value

Markov trace

Examples

tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
tm <- populate_transition_matrix(2, tmat, c(0.5, 0.5, 0, 1))
a <- health_state("Healthy", 1, 1, 0, FALSE)
b <- health_state("Dead", 1, 0, 0, TRUE)
health_states <- combine_state(a, b)
this.strategy <- strategy(tm, health_states, "intervention")
markov_model(this.strategy, 10, c(1, 0))
Usage

markov_model_sojourntime(
  current_strategy,  
  nCycles, 
  initial_state, 
  discount = c(0, 0), 
  parameter_values = NULL, 
  half_cycle_correction = TRUE, 
  state_cost_only_prevalent = FALSE, 
  state_util_only_prevalent = FALSE, 
  method = "half cycle correction", 
  startup_cost = NULL, 
  startup_util = NULL 
)

Arguments

current_strategy
  strategy object

nCycles
  no of cycles

initial_state
  value of states initially

discount
  rate of discount for costs and qalys

parameter_values
  parameters for assigning health states and probabilities

half_cycle_correction
  boolean to indicate half cycle correction

state_cost_only_prevalent
  boolean parameter to indicate if the costs for state occupancy is only for those in the state excluding those that transitioned new. This is relevant when the transition cost is provided for eg. in a state with dialysis the cost of previous dialysis is different from the newly dialysis cases. Then the state_cost_only_prevalent should be TRUE

state_util_only_prevalent
  boolean parameter to indicate if the utilities for state occupancy is only for those in the state excluding those that transitioned new.

method
  what type of half cycle correction needed

startup_cost
  cost of states initially

startup_util
  utility of states initially if any

Details

Use the strategy, cycles, initial state values creating the markov model and trace. As many probabilities /cost/utility value depend on age/time the evaluation and assignment happens during each cycle. At the heart it does a matrix multiplication using the previous row of the trace matrix and the columns of the transition matrix. Also checks for population loss, calculates cumulative costs and qalys (accounts for discounting and half cycle correction)
### microcosting_liquids_long

*Function to estimate the cost of liquids when IPD is in long format*

#### Description

Function to estimate the cost of liquids when IPD is in long format

#### Usage

```r
microcosting_liquids_long(
  the_columns,
  ind_part_data_long,
  name_med,
  brand_med = NULL,
  dose_med,
  unit_med = NULL,
  bottle_size,
  bottle_size_unit = NULL,
  bottle_lasts,
  bottle_lasts_unit = NULL,
  preparation_dose = NULL,
  preparation_unit = NULL,
  timeperiod,
  unit_cost_data,
  unit_cost_column,
  cost_calculated_per,
  strength_column,
  list_of_code_names = NULL,
  list_of_code_brand = NULL,
  list_of_code_dose_unit = NULL,
  list_of_code_bottle_size_unit = NULL,
  list_of_code_bottle_lasts_unit = NULL,
)```

#### Examples

```r
tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
tm <- populate_transition_matrix(2, tmat, c(0.5, 0.5, 0, 1))
a <- health_state("Healthy", 1, 1, 0, FALSE)
b <- health_state("Dead", 1, 0, 0, TRUE)
health_states <- combine_state(a, b)
this.strategy <- strategy(tm, health_states, "intervention")
markov_model(this.strategy, 10, c(1, 0))
```
list_preparation_dose_unit = NULL,
eqdose_covtab = NULL,
basis_strength_unit = NULL
)

Arguments

the_columns  columns that are to be used to convert the data from long to wide
ind_part_data_long
ind_part_data_long
name_med  name of medication
brand_med  brand name of medication if revealed
dose_med  dose of medication used
unit_med  unit of medication ; use null if its along with the dose
bottle_size  size of the bottle used
bottle_size_unit  unit of bottle volume
bottle_lasts  how long the bottle lasted
bottle_lasts_unit  time unit of how long the bottle lasted
preparation_dose  dose if preparation is given
preparation_unit  unit of preparatio dose
timeperiod  time period for cost calculation
unit_cost_data  unit costs data
unit_cost_column  column name of unit cost in unit_cost_data
cost_calculated_per  column name of unit where the cost is calculated
strength_column  column column name that has strength of medication
list_of_code_names  if names is coded, give the code:name pairs, optional
list_of_code_brand  if brand names are coded, give the code:brand pairs, optional
list_of_code_dose_unit  if unit is coded, give the code:unit pairs, optional
list_of_code_bottle_size_unit  list of bottle size units and codes
list_of_code_bottle_lasts_unit  list of time of bottle lasts and codes
list_preparation_dose_unit  list of preparation dose units and codes
eqdose_covtab  table to get the conversion factor for equivalent doses, optional
basis_strength_unit  strength unit to be taken as basis required for total medication calculations
Value

the calculated cost of tablets along with original data

Examples

```r
med_costs_file <- system.file("extdata", "medication_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication_liq.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx", package = "packDAMipd")
table <- load_trial_data(conv_file)
names <- colnames(ind_part_data)
ending <- length(names)
ind_part_data_long <- tidyr::gather(ind_part_data, measurement, value, names[2]:names[ending], factor_key = TRUE)
the_columns <- c("measurement", "value")
res <- microcosting_liquids_long(the_columns,
ind_part_data_long = ind_part_data_long,
name_med = "liq_name", brand_med = NULL, dose_med = "liq_strength",
unit_med = NULL, bottle_size = "liq_bottle_size", bottle_size_unit = NULL,
bottle_lasts = "liq_lasts", bottle_lasts_unit = NULL, preparation_dose = NULL,
preparation_unit = NULL, timeperiod = "4 months", unit_cost_data = med_costs,
unit_cost_column = "UnitCost", cost_calculated_per = "Basis",
strength_column = "Strength", list_of_code_names = NULL,
list_of_code_brand = NULL, list_of_code_dose_unit = NULL,
list_of_code_bottle_size_unit = NULL, list_of_code_bottle_lasts_unit = NULL,
list_preparation_dose_unit = NULL, eqdose_covtab = table,
basis_strength_unit = NULL)
```

---

**Function to estimate the cost of liquids taken (from IPD)**

**Description**

Function to estimate the cost of liquids taken (from IPD)

**Usage**

```r
microcosting_liquids_wide(
  ind_part_data,
  name_med,
  brand_med = NULL,
  dose_med,
  unit_med = NULL,
  bottle_size,
```
Arguments

ind_part_data IPD
name_med name of medication
brand_med brand name of medication if revealed
dose_med dose of medication used
unit_med unit of medication; use null if it's along with the dose
bottle_size size of the bottle used
bottle_size_unit unit of bottle volume
bottle_lasts how long the bottle lasted
bottle_lasts_unit time unit of how long the bottle lasted
preparation_dose dose if preparation is given
preparation_unit unit of preparation dose
timeperiod time period for cost calculation
unit_cost_data unit costs data
unit_cost_column column name of unit cost in unit_cost_data
cost_calculated_per column name of unit where the cost is calculated
strength_column column name that has strength of medication
microcosting_liquids_wide

- list_of_code_names
  - if names is coded, give the code:name pairs, optional
- list_of_code_brand
  - if brand names are coded, give the code:brand pairs, optional
- list_of_code_dose_unit
  - if unit is coded, give the code:unit pairs, optional
- list_of_code_bottle_size_unit
  - list of bottle size units and codes
- list_of_code_bottle_lasts_unit
  - list of time of bottle lasts and codes
- list_preparation_dose_unit
  - list of preparation dose units and codes
- eqdose_covtab
  - table to get the conversion factor for equivalent doses, optional, but the column names have to unique Similar to c("Drug", "form", "unit", "factor") or c("Drug", "form", "unit", "conversion")
- basis_strength_unit
  - strength unit to be taken as basis required for total medication calculations

Value

- the calculated cost of tablets along with original data

Examples

```r
med_costs_file <- system.file("extdata", "medicaton_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication_liq.xlsx", package = "packDAMipd")
ind_part_data <- loadTrialData(data_file)
med_costs <- loadTrialData(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx", package = "packDAMipd")
table <- loadTrialData(conv_file)
res <- microcosting_liquids_wide(
  ind_part_data = ind_part_data, name_med = "liq_name", brand_med = NULL,
dose_med = "liq_strength", unit_med = NULL, bottle_size = "liq_bottle_size",
bottle_size_unit = NULL, bottle_lasts = "liq_lasts",
bottle_lasts_unit = NULL, preparation_dose = NULL, preparation_unit = NULL,
timeperiod = "4 months", unit_cost_data = med_costs,
unit_cost_column = "UnitCost", cost_calculated_per = "Basis",
strength_column = "Strength", list_of_code_names = NULL,
list_of_code_brands = NULL, list_of_code_dose_unit = NULL,
list_of_code_bottle_size_unit = NULL, list_of_code_bottle_lasts_unit = NULL,
list_preparation_dose_unit = NULL, eqdose_covtab = table,
basis_strength_unit = NULL)
```
Function to estimate the cost of patches when IPD is in long format using a IPD data of long format

Usage

microcosting_patches_long(
  the_columns,
  ind_part_data_long, 
  name_med, 
  brand_med = NULL, 
  dose_med, 
  unit_med = NULL, 
  no_taken, 
  freq_taken, 
  timeperiod, 
  unit_cost_data, 
  unit_cost_column, 
  cost_calculated_per, 
  strength_column, 
  list_of_code_names = NULL, 
  list_of_code_freq = NULL, 
  list_of_code_dose_unit = NULL, 
  list_of_code_brand = NULL, 
  eqdose_cov_tab = NULL, 
  basis_strength_unit = NULL
)
freq_taken  frequency of medication
timeperiod   time period for cost calculation
unit_cost_data  unit costs data
unit_cost_column column name of unit cost in unit_cost_data
cost_calculated_per column name of unit in the cost is calculated
strength_column column name that contain strength of medication
list_of_code_names if names is coded, give the code:name pairs, optional
list_of_code_freq if frequency is coded, give the code:frequency pairs, optional
list_of_code_dose_unit if unit is coded, give the code:unit pairs, optional
list_of_code_brand if brand names are coded, give the code:brand pairs, optional
eqdose_cov_tab table to get the conversion factor for equivalent doses, optional
basis_strength_unit strength unit to be taken as basis required for total medication calculations

Value
the calculated cost of tablets along with original data

Examples
med_costs_file <- system.file("extdata", "medicaton_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx",package = "packDAMipd")
table <- load_trial_data(conv_file)
names <- colnames(ind_part_data)
ending <- length(names)
ind_part_data_long <- tidyr::gather(ind_part_data, measurement, value, names[2]:names[ending], factor_key = TRUE)
the_columns <- c("measurement", "value")
res <- microcosting_patches_long(the_columns,
ind_part_data_long = ind_part_data_long, name_med = "patch_name", brand_med = "patch_brand", dose_med = "patch_strength",unit_med = NULL, no_taken = "patch_no_taken", freq_taken = "patch_frequency",
timeperiod = "4 months", unit_cost_data = med_costs,
unit_cost_column = "UnitCost", cost_calculated_per = "Basis", strength_column = "Strength", list_of_code_names = NULL,
list_of_code_freq = NULL, list_of_code_dose_unit = NULL,
list_of_code_brand = NULL, eqdose_cov_tab = table,
basis_strength_unit = "mcg/hr")
Function to estimate the cost of patches taken (from IPD)

**Description**

Function to estimate the cost of patches taken (from IPD)

**Usage**

```r
microcosting_patches_wide(
    ind_part_data,
    name_med,
    brand_med = NULL,
    dose_med,
    unit_med = NULL,
    no_taken,
    freq_taken,
    timeperiod,
    unit_cost_data,
    unit_cost_column,
    cost_calculated_per,
    strength_column,
    list_of_code_names = NULL,
    list_of_code_freq = NULL,
    list_of_code_dose_unit = NULL,
    list_of_code_brand = NULL,
    eqdose_cov_tab = NULL,
    basis_strength_unit = NULL
)
```

**Arguments**

- **ind_part_data**: IPD
- **name_med**: name of medication
- **brand_med**: brand name of medication if revealed
- **dose_med**: dose of medication used
- **unit_med**: unit of medication; use null if its along with the dose
- **no_taken**: how many taken
- **freq_taken**: frequency of medication
- **timeperiod**: time period for cost calculation
- **unit_cost_data**: unit costs data
- **unit_cost_column**: column name of unit cost in unit_cost_data
cost_calculated_per
column name of unit where the cost is calculated

strength_column
column column name that contain strength of medication

list_of_code_names
if names is coded, give the code:name pairs, optional

list_of_code_freq
if frequency is coded, give the code:frequency pairs, optional

list_of_code_dose_unit
if unit is coded, give the code:unit pairs, optional

list_of_code_brand
if brand names are coded, give the code:brand pairs, optional

eqdose_cov_tab
table to get the conversion factor for equivalent doses, optional, but the column names have to unique Similar to c("Drug", "form", "unit", "factor") or c("Drug", "form", "unit", "conversion")

basis_strength_unit
strength unit to be taken as basis required for total medication calculations

Details
Assumes individual level data has name of medication, dose, dose unit, number taken, frequency taken, and basis time Assumes unit cost data contains the name of medication, form/type, strength, unit of strength (or the unit in which the cost calculated), preparation, unit cost, size and size unit (in which name, forms, size, size unit, and preparation are not passed on) @importFrom dplyr %>% a patient use 1 mg/hr patches 5 patches once a week that patch comes in a pack of 4 with cost £2.50 we want to estimate the cost for 3 months that means amount of medication 3 months = 21 weeks number of patches taken = 215 = 105 patches packs = (105/4) almost 27 packs cost = 27.50

Value
the calculated cost of tablets along with original data

Examples

```r
med_costs_file <- system.file("extdata", "medication_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx", package = "packDAMipd")
table <- load_trial_data(conv_file)
res <- microcosting_patches_wide(
  ind_part_data = ind_part_data, name_med = "patch_name",
  brand_med = "patch_brand", dose_med = "patch_strength", unit_med = NULL,
  no_taken = "patch_no_taken", freq_taken = "patch_frequency",
  timeperiod = "4 months", unit_cost_data = med_costs, unit_cost_column = "UnitCost", cost_calculated_per = "Basis",
  strength_column = "Strength", list_of_code_names = NULL,
)```

Function to estimate the cost of tablets when IPD is in long format

**Description**

Function to estimate the cost of tablets when IPD is in long format

**Usage**

```r
microcosting_tablets_long(
  the_columns,
  ind_part_data_long,
  name_med,
  brand_med = NULL,
  dose_med,
  unit_med = NULL,
  no_taken,
  freq_taken,
  timeperiod,
  unit_cost_data,
  unit_cost_column,
  cost_calculated_per,
  strength_column,
  list_of_code_names = NULL,
  list_of_code_freq = NULL,
  list_of_code_dose_unit = NULL,
  list_of_code_brand = NULL,
  eqdose_cov_tab = NULL,
  basis_strength_unit = NULL
)
```

**Arguments**

- `the_columns`: columns that are to be used to convert the data from long to wide IPD
- `ind_part_data_long`: IPD
- `name_med`: name of medication
- `brand_med`: brand name of medication if revealed
- `dose_med`: dose of medication used
- `unit_med`: unit of medication; use null if its along with the dose
- `no_taken`: how many taken
freq_taken  
  frequency of medication

timeperiod  
  time period for cost calculation

unit_cost_data  
  unit costs data

unit_cost_column  
  column name of unit cost in unit_cost_data

cost_calculated_per  
  column name of unit where the cost is calculated

strength_column  
  column column name that contain strength of medication

list_of_code_names  
  if names is coded, give the code:name pairs, optional

list_of_code_freq  
  if frequency is coded, give the code:frequency pairs, optional

list_of_code_dose_unit  
  if unit is coded, give the code:unit pairs, optional

list_of_code_brand  
  if brand names are coded, give the code:brand pairs, optional

eqdose_cov_tab  
  table to get the conversion factor for equivalent doses, optional

basis_strength_unit  
  strength unit to be taken as basis required for total medication calculations

Value

the calculated cost of tablets along with original data

Examples

```r
med_costs_file <- system.file("extdata", "medicaton_costs_all.xlsx",
package = "packDAMipd")
data_file <- system.file("extdata", "medication_all.xlsx",
package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx", package = "packDAMipd")
table <- load_trial_data(conv_file)
names <- colnames(ind_part_data)
ending <- length(names)
ind_part_data_long <- tidyr::gather(ind_part_data, measurement, value,
  names[2]:names[ending], factor_key = TRUE)
the_columns <- c("measurement", "value")
res <- microcosting_tablets_long(the_columns,
  ind_part_data_long = ind_part_data_long, name_med = "tab_name",
  brand_med = "tab_brand", dose_med = "tab_strength",
  unit_med = "tab_str_unit",
  no_taken = "tab_no_taken", freq_taken = "tab_frequency",
  timeperiod = "2 months", unit_cost_data = med_costs,
  unit_cost_column = "UnitCost", cost_calculated_per = "Basis",
  strength_column = "Strength", list_of_code_names = NULL,
  list_of_code_freq = NULL, list_of_code_dose_unit = NULL,
eqdose_cov_tab = table, basis_strength_unit = "mg")
```
Function to estimate the cost of tablets taken (from IPD)

Description

Function to estimate the cost of tablets taken (from IPD)

Usage

```r
microcosting_tablets_wide(
  ind_part_data,
  name_med,
  brand_med = NULL,
  dose_med,
  unit_med = NULL,
  no_taken,
  freq_taken,
  timeperiod,
  unit_cost_data,
  unit_cost_column,
  cost_calculated_per,
  strength_column,
  list_of_code_names = NULL,
  list_of_code_freq = NULL,
  list_of_code_dose_unit = NULL,
  list_of_code_brand = NULL,
  eqdose_cov_tab = NULL,
  basis_strength_unit = NULL
)
```

Arguments

- `ind_part_data` : IPD
- `name_med` : name of medication
- `brand_med` : brand name of medication if revealed
- `dose_med` : dose of medication used
- `unit_med` : unit of medication; use null if it's along with the dose
- `no_taken` : how many taken
- `freq_taken` : frequency of medication
- `timeperiod` : time period for cost calculation
- `unit_cost_data` : unit costs data
- `unit_cost_column` : column name of unit cost in unit_cost_data
cost_calculated_per
  column name of unit where the cost is calculated
strength_column
  column column name that contain strength of medication
list_of_code_names
  if names is coded, give the code:name pairs, optional
list_of_code_freq
  if frequency is coded, give the code:frequency pairs, optional
list_of_code_dose_unit
  if unit is coded, give the code:unit pairs, optional
list_of_code_brand
  if brand names are coded, give the code:brand pairs, optional
eqdose_cov_tab
  table to get the conversion factor for equivalent doses, optional, but the column names have to unique
  Similar to c("Drug", "form", "unit", "factor") or c("Drug", "form", "unit", "conversion")
basis_strength_unit
  strength unit to be taken as basis required for total medication calculations

Details

Assumes individual level data has name of medication, dose, dose unit, number taken, frequency taken, and basis time
Assumes unit cost data contains the name of medication, form/type, strength, unit of strength (or the unit in which the cost calculated), preparation, unit cost, size and size unit (in which name, forms, size, size unit, and preparation are not passed on)

Value

the calculated cost of tablets along with original data

Examples

med_costs_file <- system.file("extdata", "medication_costs_all.xlsx", package = "packDAMipd")
data_file <- system.file("extdata", "medication_all.xlsx", package = "packDAMipd")
ind_part_data <- load_trial_data(data_file)
med_costs <- load_trial_data(med_costs_file)
conv_file <- system.file("extdata", "Med_calc.xlsx", package = "packDAMipd")
table <- load_trial_data(conv_file)
res <- microcosting_tablets_wide(ind_part_data = ind_part_data, name_med = "tab_name", brand_med = "tab_brand", dose_med = "tab_strength", unit_med = "tab_str_unit", no_taken = "tab_no_taken", freq_taken = "tab_frequency", timeperiod = "2 months", unit_cost_data = med_costs, unit_cost_column = "UnitCost", cost_calculated_per = "Basis", strength_column = "Strength", list_of_code_names = NULL, list_of_code_freq = NULL, list_of_code_dose_unit = NULL, eqdose_cov_tab = table, basis_strength_unit = "mg")
**plot_ceac_psa**  
*Function to plot CEAC*

**Description**

Function to plot CEAC

**Usage**

```
plot_ceac_psa(
  control_markov,
  trt_markov,
  psa_table_ctrl,
  psa_table_trt,
  thresholds,
  num_rep,
  comparator
)
```

**Arguments**

- `control_markov`: markov model for control
- `trt_markov`: markov model for treatment
- `psa_table_ctrl`: param table for psa for control
- `psa_table_trt`: param table for psa for treatment
- `thresholds`: threshold values of WTP
- `num_rep`: number of repetitions
- `comparator`: the strategy to be compared with

**Value**

`plot of ceac`

---

**plot_dsa**  
*Function to plot results of sensitivity analysis do_sensitivity_analysis()*

**Description**

Function to plot results of sensitivity analysis `do_sensitivity_analysis()`
Usage

```r
plot_dsa(
  result_dsa_control,
  plotfor,
  type = "range",
  result_dsa_treat = NULL,
  threshold = NULL,
  comparator = NULL
)
```

Arguments

- `result_dsa_control`: result from deterministic sensitivity analysis for first or control model
- `plotfor`: the variable to plot for e.g. cost, utility NMB etc
- `type`: type of analysis, range or difference
- `result_dsa_treat`: result from deterministic sensitivity analysis for the comparative Markov model
- `threshold`: threshold value of WTP
- `comparator`: the strategy to be compared with

Value

- plot of sensitivity analysis

Examples

```r
param_list <- define_parameters(
  cost_zido = 2278, cost_direct_med_A = 1701,
  cost_comm_care_A = 1055, cost_direct_med_B = 1774,
  cost_comm_care_B = 1278,
  cost_direct_med_C = 6948, cost_comm_care_C = 2059,
  tpAtoA = 1251 / (1251 + 483),
  tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
  tpAtoD = 17 / (17 + 1717),
  tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
  tpBtoD = 15 / (15 + 1243),
  tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312),
  tpDtoD = 1,
  cost_health_A = "cost_direct_med_A + cost_comm_care_A",
  cost_health_B = "cost_direct_med_B + cost_comm_care_B",
  cost_health_C = "cost_direct_med_C + cost_comm_care_C",
  cost_drug = "cost_zido")
low_values <- define_parameters(cost_direct_med_B = 177.4,
  cost_comm_care_C = 205.9)
upp_values <- define_parameters(cost_direct_med_B = 17740,
  cost_comm_care_C = 20590)
A <- health_state("A", cost = "cost_health_A + cost_drug ",
```
utility = 1
B <- health_state("B", cost = "cost_health_B + cost_drug",
 utility = 1)
C <- health_state("C", cost = "cost_health_C + cost_drug",
 utility = 1)
D <- health_state("D", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3, 4),
 c(NA, 5, 6, 7), c(NA, NA, 8, 9),
 c(NA, NA, NA, 10))
colnames(tmat) <- rownames(tmat) <- c("A", "B", "C", "D")
tm <- populate_transition_matrix(4, tmat, c("tpAtoA", "tpAtoB", "tpAtoC",
 "tpAtoD", "tpBtoB", "tpBtoC", "tpBtoD", "tpCtoC", "tpCtoD", "tpDtoD"),
colnames(tmat))
health_states <- combine_state(A, B, C, D)
mono_strategy <- strategy(tm, health_states, "mono")
mono_markov <- markov_model(mono_strategy, 20,
 c(1, 0, 0, 0),
 discount = c(0.06, 0), param_list)
param_table <- define_parameters_sens_anal(param_list, low_values,
 upp_values)
result <- do_sensitivity_analysis(mono_markov, param_table)
param_list_treat <- define_parameters(
 cost_zido = 3000, cost_direct_med_A = 890,
 cost_comm_care_A = 8976, cost_direct_med_B = 2345,
 cost_comm_care_B = 1278,
 cost_direct_med_C = 6948, cost_comm_care_C = 2059,
 tpAtoA = 1251 / (1251 + 483),
 tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
 tpAtoD = 17 / (17 + 1717),
 tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
 tpBtoD = 15 / (15 + 1243),
 tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312),
 tpDtoD = 1,
 cost_health_A = "cost_direct_med_A + cost_comm_care_A",
 cost_health_B = "cost_direct_med_B + cost_comm_care_B",
 cost_health_C = "cost_direct_med_C + cost_comm_care_C",
 cost_drug = "cost_zido")
treat_strategy <- strategy(tm, health_states, "treat")
treat_markov <- markov_model(treat_strategy, 20,
 c(1, 0, 0, 0),
 discount = c(0.06, 0), param_list_treat)
treat_low_values <- define_parameters(cost_direct_med_B = 234.5,
 cost_comm_care_C = 694.8)
treat_upp_values <- define_parameters(cost_direct_med_B = 23450,
 cost_comm_care_C = 69480)
param_table_treat <- define_parameters_sens_anal(param_list_treat,
 treat_low_values, treat_upp_values)
result_treat <- do_sensitivity_analysis(treat_markov, param_table)
plot_dsa(result, "NMB", "range", result_treat, 20000, "treat")
**plot_dsa_icer_range**

**Description**
Function to do some checks before plotting sensitivity analysis results

**Usage**

```r
plot_dsa_icer_range(ob_results, plot_for, plot_var)
```

**Arguments**

- `ob_results`: results from deterministic sensitivity analysis
- `plot_for`: the quantity plotting
- `plot_var`: the variable

**Value**

- `plot`
**plot_dsa_nmb_range**  
*Function to do some checks before plotting sensitivity analysis results*

**Description**
Function to do some checks before plotting sensitivity analysis results

**Usage**

```r
plot_dsa_nmb_range(ob_results, plot_var)
```

**Arguments**
- `ob_results`: results from deterministic sensitivity analysis
- `plot_var`: the variable

**Value**
plot

---

**plot_dsa_others_range**  
*Function to do some checks before plotting sensitivity analysis results*

**Description**
Function to do some checks before plotting sensitivity analysis results

**Usage**

```r
plot_dsa_others_range(ob_results, plot_var)
```

**Arguments**
- `ob_results`: results from deterministic sensitivity analysis
- `plot_var`: the variable

**Value**
plot
plot_efficiency_frontier

Plot efficiency frontier

Description

Plot efficiency frontier

Usage

plot_efficiency_frontier(results_calculate_icer_nmb, threshold)

Arguments

results_calculate_icer_nmb
results from cca (from calculate_icer_nmb method)

threshold
threshold value

Value

plot well <- health_state("well", cost = 0, utility = 1) disabled <- health_state("disabled", cost = 100, utility = 1) dead <- health_state("dead", cost = 0, utility = 0) tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6)) colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead") tm <- populate_transition_matrix(3, tmat, c(0.6, 0.2, 0.2, 0.6, 0.4, 1), colnames(tmat)) health_states <- combine_state(well, disabled, dead) this.strategy <- strategy(tm, health_states, "control") this_markov <- markov_model(this.strategy, 24, c(1000, 0, 0), c(0,0)) well <- health_state("well", cost = 0, utility = 1) disabled <- health_state("disabled", cost = 10, utility = 0.5) dead <- health_state("dead", cost = 0, utility = 0) tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6)) colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead") tm <- populate_transition_matrix(3, tmat, c(0.4, 0.4, 0.2, 0.6, 0.4, 1), colnames(tmat)) health_states <- combine_state(well, disabled, dead) this.strategy <- strategy(tm, health_states, "intervention") sec_markov <- markov_model(this.strategy, 24, c(1000, 0, 0), c(0,0)) list_markov <- combine_markov(this_markov, sec_markov) results_cea <- calculate_icer_nmb(list_markov, 20000, comparator = "control") plot_efficiency_frontier(results_cea, c(1000, 2000))

plot_histogram_onetimepoint_twogroups

Function to plot mean and SE for longitudinal observations for twogroups compared

Description

Function to plot mean and SE for longitudinal observations for twogroups compared
plot_meanSE_longitudinal_twogroups

Function to plot mean and SE for longitudinal observations for twogroups compared

Description

Function to plot mean and SE for longitudinal observations for twogroups compared
plot_model

Usage

plot_meanSE_longitudinal_twogroups(
    thedata,
    columnnames,
    timepoints,
    observation
)

Arguments

thedata  the data where the observations are held
columnnames columnnames in the data where the interested observations
            timepoints the timepoints at which the descriptive analysis is done
observation name of the observations

Value

the plot that shows mean and SE

Examples

eg_data <- as.data.frame(list(no = c(1, 2, 3, 4),
      mark_at_1 = c(12, 7, 23, 45), gender = c("M", "F", "M", "F"),
      mark_at_2 = c(12, 34, 89, 45), trialarm = c("1","1","2","2")))
plot_meanSE_longitudinal_twogroups(eg_data, c("mark_at_1", "mark_at_2"), c("1","2"), "mark")

plot_model

E1. Plot a Markov model

Description

E1. Plot a Markov model

Usage

plot_model(markov)

Arguments

markov markov_model object

Value

plots
plot_nmb_lambda

Examples

tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
tm <- populate_transition_matrix(2, tmat, c(0.5, 0.5, 0, 1))
a <- health_state("Healthy", 1, 1, 0, FALSE)
b <- health_state("Dead", 1, 0, 0, TRUE)
health_states <- combine_state(a, b)
this.strategy <- strategy(tm, health_states, "intervention")
this_markov <- markov_model(this.strategy, 10, c(1, 0), c(0, 0))
p <- plot_model(this_markov)

plot_nmb_lambda

Plot cost effectiveness acceptability curve

Description

Plot cost effectiveness acceptability curve

Usage

plot_nmb_lambda(list_markov, threshold_values, comparator, currency = "GBP")

Arguments

list_markov markov_model objects
threshold_values list of threshold values
comparator the comparator
currency currency

Value

plots

Examples

well <- health_state("well", cost = 0, utility = 1)
disabled <- health_state("disabled", cost = 100, utility = 1)
dead <- health_state("dead", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead")
tm <- populate_transition_matrix(3, tmat, c(0.6, 0.2, 0.2, 0.6, 0.4, 1),
colnames(tmat))
health_states <- combine_state(well, disabled, dead)
this.strategy <- strategy(tm, health_states, "control")
this_markov <- markov_model(this.strategy, 24, c(1000, 0, 0), c(0, 0))
well <- health_state("well", cost = 0, utility = 1)
disabled <- health_state("disabled", cost = 10, utility = 0.5)
dead <- health_state("dead", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("well", "disabled", "dead")
tm <- populate_transition_matrix(3, tmat, c(0.4, 0.4, 0.2, 0.6, 0.4, 1),
colnames(tmat))
health_states <- combine_state(well, disabled, dead)
this.strategy <- strategy(tm, health_states, "intervention")
sec_markov <- markov_model(this.strategy, 24, c(1000, 0, 0), c(0, 0))
list_markov <- combine_markov(this_markov, sec_markov)
plot_nmb_lambda(list_markov, c(1000, 2000, 3000), comparator = "control")

---

**plot_prediction_parametric_survival**

*Plot the predicted survival curves for covariates keeping the others fixed*

**Description**

Plot the predicted survival curves for covariates keeping the others fixed

**Usage**

```r
plot_prediction_parametric_survival(
  param_to_be_estimated,
  indep_var,
  covariates,
  dataset,
  fit,
  timevar_survival
)
```

**Arguments**

- `param_to_be_estimated`  
  parameter to be estimated
- `indep_var`  
  variable for which the levels have to be identified
- `covariates`  
  the covariates
- `dataset`  
  the dataset where these variables contain
- `fit`  
  the fit result survreg
- `timevar_survival`  
  time variable from the dataset
**plot_return_residual_cox**

Value

plot

Examples

```r
data_for_survival <- survival::lung
surv_estimated <- use_parametric_survival("status", data_for_survival, "sex", info_distribution = "weibull", covariates = c("ph.ecog","time")
plot_prediction_parametric_survival("status", "sex", covariates = c("ph.ecog"), data_for_survival, surv_estimated$fit, "time")
```

**Description**

Plotting and return the residuals after cox proportional hazard model

**Usage**

```r
plot_return_residual_cox(
  param_to_be_estimated,
  indep_var,
  covariates,
  fit,
  dataset
)
```

Arguments

- `param_to_be_estimated`: parameter to be estimated
- `indep_var`: independent variable
- `covariates`: covariates
- `fit`: fit object from coxph method
- `dataset`: data used for cox ph model

**Value**

plot and the residuals
plot_return_residual_survival

Examples

```r
data_for_survival <- survival::lung
surv_estimated <- use_coxph_survival("status", data_for_survival, "sex", 
covariates = c("ph.ecog"), "time")
plot_return_residual_cox("status", "sex", covariates = c("ph.ecog"), 
surv_estimated$fit, data_for_survival )
```

---

**Description**

Plotting and return the residuals after survival model

**Usage**

```r
plot_return_residual_survival(
  param_to_be_estimated, 
  indep_var, 
  covariates, 
  fit
)
```

**Arguments**

- `param_to_be_estimated` : parameter to be estimated
- `indep_var` : independent variable
- `covariates` : covariates
- `fit` : fit object from survreg method

**Value**

plot and the residuals

**Examples**

```r
data_for_survival <- survival::lung
surv_estimated <- use_parametric_survival("status", data_for_survival, 
  "sex", 
  info_distribution = "weibull", covariates = c("ph.ecog"), "time")
plot_return_residual_survival("status", "sex", 
covariates = c("ph.ecog"), surv_estimated$fit)
```
plot_return_survival_curve

Plotting survival function for all covariates using survfit

Description

Plotting survival function for all covariates using survfit

Usage

plot_return_survival_curve(
  param_to_be_estimated,
  dataset,
  indep_var,
  covariates,
  timevar_survival
)

Arguments

param_to_be_estimated
  parameter to be estimated

dataset
  param describing the methods

indep_var
  independent variable

covariates
  covariates

timevar_survival
  time variable for survival analysis

Value

plot and the survival function values

Examples

data_for_survival <- survival::lung
plot_return_survival_curve(param_to_be_estimated = "status",
  dataset = data_for_survival,indep_var = "sex",covariates = c("ph.ecog"),
  timevar_survival = "time")
plot_survival_cox_covariates

Plotting survival function for all covariates calculated from cox regression results and returned coefficients

Description

Plotting survival function for all covariates calculated from cox regression results and returned coefficients

Usage

plot_survival_cox_covariates(
  coxfit, 
  dataset, 
  param_to_be_estimated, 
  covariates, 
  indep_var
)

Arguments

coxfit cox regression fit result
dataset param describing the methods
param_to_be_estimated parameter to be estimated
covariates covariates
indep_var independent variable

Value

plot and the survival function values

Examples

data_for_survival <- survival::lung
surv_estimated <- use_coxph_survival("status", data_for_survival, "sex", 
covariates = c("ph.ecog"), "time")
plot_survival_cox_covariates(surv_estimated$fit, data_for_survival, 
"status", covariates = c("ph.ecog"), "sex")
Description

Populate transition matrix

Usage

populate_transition_matrix(no_states, tmat, list_prob, name_states = NULL)

Arguments

no_states  number of the health states
tmat       A transition matrix in the format from the package 'mstate'
list_prob  list of probabilities as in the order of transitions (row wise)
name_states  names of the health states

Details

If the state names are null, they are replaced with numbers starting from 1 First find those missing probabilities, and fill a list from the given list of probabilities and fill those are not NA in the matrix Note that the probabilities need not be numeric here and no checks are needed for sum

Value

value of the transition matrix

Examples

tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
populate_transition_matrix(2, tmat, list_prob = c(0.2, 0.5, 0, 0.3))

predict_coxph  Predict risk/hazard function for cox ph regression

Description

Predict risk/hazard function for cox ph regression
Usage

predict_coxph(
  coxfit,
  dataset,
  param_to_be_estimated,
  covariates,
  indep_var,
  timevar_survival
)

Arguments

coxfit cox regression fit result
dataset param describing the methods
param_to_be_estimated parameter to be estimated
covariates covariates
indep_var independent variable
timevar_survival time variable

Details

"risk" option for "type" returns the hazard ratio relative to mean e.g given below For lung data with

data_for_survival <- survival::lung
fit <- use_coxph_survival("status", data_for_survival, "sex", covariates = c("ph.ecog"), "time")
coeffit = fit$coefficients
r1234 <- exp(coefit("sex")lung$sex+coeffit("ph.ecog")lung$ph.ecog)
rMean <- exp(sum(coef(fit) * fit$means, na.rm=TRUE))
rr <- r1234/rMean

Value

plot and the survival function values

Examples

data_for_survival <- survival::lung
surv_estimated <- use_coxph_survival("status", data_for_survival,
  "sex",covariates = c("ph.ecog"), "time")
predict_coxph(surv_estimated$fit,data_for_survival, "status","sex",
covariates = c("ph.ecog"), "time")
**promis3a_scoring.df**

**Description**

promis 3a scoring table

**Usage**

promis3a_scoring.df

**Format**

A 14 by 3 dataframe

**Source**

created on April 08, 2021

---

**report_sensitivity_analysis**

*Function to report deterministic sensitivity analysis*

**Description**

Function to report deterministic sensitivity analysis

**Usage**

report_sensitivity_analysis(
    result_dsa_control,
    result_dsa_treat = NULL,
    threshold = NULL,
    comparator = NULL
)

**Arguments**

- `result_dsa_control`
  - result from deterministic sensitivity analysis for first or control model
- `result_dsa_treat`
  - result from deterministic sensitivity analysis for the comparative Markov model
- `threshold`
  - threshold value of WTP
- `comparator`
  - the strategy to be compared with
Value

report in the form of a table

Examples

```r
param_list <- define_parameters(
  cost_zido = 2278, cost_direct_med_A = 1701,
  cost_comm_care_A = 1055, cost_direct_med_B = 1774,
  cost_comm_care_B = 1278,
  cost_direct_med_C = 6948, cost_comm_care_C = 2059,
  tpAtoA = 1251 / (1251 + 483),
  tpAtoB = 350 / (350 + 1384), tpAtoC = 116 / (116 + 1618),
  tpAtoD = 17 / (17 + 1717),
  tpBtoB = 731 / (731 + 527), tpBtoC = 512 / (512 + 746),
  tpBtoD = 15 / (15 + 1243),
  tpCtoC = 1312 / (1312 + 437), tpCtoD = 437 / (437 + 1312),
  tpDtoD = 1,
  cost_health_A = "cost_direct_med_A + cost_comm_care_A",
  cost_health_B = "cost_direct_med_B + cost_comm_care_B",
  cost_health_C = "cost_direct_med_C + cost_comm_care_C",
  cost_drug = "cost_zido")
low_values <- define_parameters(cost_direct_med_B = 177.4,
  cost_comm_care_C = 205.9)
upp_values <- define_parameters(cost_direct_med_B = 17740,
  cost_comm_care_C = 20590)
A <- health_state("A", cost = "cost_health_A + cost_drug ", utility = 1)
B <- health_state("B", cost = "cost_health_B + cost_drug", utility = 1)
C <- health_state("C", cost = "cost_health_C + cost_drug", utility = 1)
D <- health_state("D", cost = 0, utility = 0)
tmat <- rbind(c(1, 2, 3, 4), c(NA, 5, 6, 7), c(NA, NA, 8, 9),
  c(NA, NA, NA, 10))
colnames(tmat) <- rownames(tmat) <- c("A", "B", "C", "D")
tm <- populate_transition_matrix(4, tmat, c("tpAtoA", "tpAtoB", "tpAtoC",
  "tpAtoD", "tpBtoB", "tpBtoC", "tpBtoD", "tpCtoC", "tpCtoD", "tpDtoD"),
colnames(tmat))
health_states <- combine_state(A, B, C, D)
mono_strategy <- strategy(tm, health_states, "mono")
mono_markov <- markov_model(mono_strategy, 20, c(1, 0, 0, 0),
  discount = c(0.06, 0), param_list)
param_table <- define_parameters_sens_anal(param_list, low_values,
  upp_values)
result <- do_sensitivity_analysis(mono_markov, param_table)
reporting <- report_sensitivity_analysis(result)
```

return0_if_not_null_na

Function to return 0 if the param is not null or NA trimming the white spaces
**return_equal_liststring_col**

Function to get the subset of data compared to a string after trimming the white spaces

**Usage**

```
return_equal_liststring_col(col, the_data, list_str)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>col</td>
<td>the form of medication either tablet or patch</td>
</tr>
<tr>
<td>the_data</td>
<td>the data to be get the subset from</td>
</tr>
<tr>
<td>list_str</td>
<td>list of strings to be compared</td>
</tr>
</tbody>
</table>

**Value**

the subset data
Examples

```r
the_data <- as.data.frame(cbind(c("one", "two"), c("a", "b")))
colnames(the_data) <- c("name", "brand")
ans <- return_equal_liststring_listcol(2, the_data, c("a", "cc"))
```

```
return_equal_liststring_listcol
  Function to get the subset of data compared to a string after trimming the white spaces
```

Description

Function to get the subset of data compared to a string after trimming the white spaces

Usage

```r
return_equal_liststring_listcol(col, the_data, list_str)
```

Arguments

- `col` the form of medication either tablet or patch
- `the_data` the data to be get the subset from
- `list_str` list of strings to be compared

Value

the subset data

Examples

```r
the_data <- as.data.frame(cbind(c("one", "two"), c("tablet", "tablets"), c("aa", "bb")))
colnames(the_data) <- c("name", "brand_a", "xx")
ans <- return_equal_liststring_listcol(2, the_data, c("tablet", "tablets"))
```
**return_equal_str_col**

*Function to get the subset of data compared to a string after trimming the white spaces*

**Description**

Function to get the subset of data compared to a string after trimming the white spaces

**Usage**

```r
define_returns_string_col(col, the_data, the_str)
```

**Arguments**

- `col` the form of medication either tablet or patch
- `the_data` the data to be get the subset from
- `the_str` the string to be compared

**Value**

the subset data

**Examples**

```r
the_data <- as.data.frame(cbind(c("one", "two"), c("a", "b")))
colnames(the_data) <- c("name", "brand")
ans <- return_equal_str_col(2, the_data, "a")
```

**set_var_state**

*Set the attribute for the health state*

**Description**

Set the attribute for the health state

**Usage**

```r
set_var(state, var, new_value)
```

**Arguments**

- `state` object of class health state
- `var` attribute of the health state
- `new_value` new value to be assigned
Details

After checking the given state is a health state the value of the variable is set if the value is not numeric, it is being parsed to form an expression

Value

modified health state

Examples

set_var_state(health_state("IT", 100, 0.4, 0, FALSE), "cost", 1)

strategy

Definition of strategy - or arm

Description

Definition of strategy - or arm

Usage

strategy(trans_mat, states, name, trans_cost = NULL, trans_util = NULL)

Arguments

trans_mat  transition matrix
states     health states
name       name of the strategy
trans_cost values of costs if these are attached to transitions
trans_util values of utility if these are attached to transitions

Details

Defining strategy keeping all transition matrix, states and names together to use in defining Markov model

Value

object strategy

Examples

tmat <- rbind(c(1, 2), c(3, 4))
colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead")
tm <- populate_transition_matrix(2, tmat, c(0.5, 0.5, 0, 1))
a <- health_state("Healthy", 1, 1, 0, FALSE)
b <- health_state("Dead", 1, 0.5, 0, FALSE)
states <- combine_state(a, b)
strategy(tm, states, "intervention")
Function to summarise and plot probabilistic sensitivity analysis

Usage

summary_plot_psa(
  result_psa_params_control,
  result_psa_params_treat = NULL,
  threshold = NULL,
  comparator = NULL
)

Arguments

result_psa_params_control
  result from probabilistic sensitivity analysis for first or control model

result_psa_params_treat
  result from probabilistic sensitivity analysis for the comparative Markov model

threshold
  threshold value of WTP

comparator
  the strategy to be compared with

Value

plot of sensitivity analysis

Examples

param_list <- define_parameters(
  cost_direct_med_A = 1701,
  cost_direct_med_B = 1774, tpAtoA = 0.2,
  tpAtoB = 0.5, tpAtoC = 0.3,
  tpBtoB = 0.3, tpBtoC = 0.7,
  tpCtoC = 1, cost_health_A = "cost_direct_med_A",
  cost_health_B = "cost_direct_med_B"
)
sample_list <- define_parameters(cost_direct_med_A = "gamma(mean = 1701, 
  sd = sqrt(1701))")
A <- health_state("A", cost = "cost_health_A ", utility = 1)
B <- health_state("B", cost = "cost_health_B", utility = 1)
C <- health_state("C", cost = 0, utility = 0, absorb = "TRUE")
tmat <- rbind(c(1, 2, 3), c(NA, 4, 5), c(NA, NA, 6))
colnames(tmat) <- rownames(tmat) <- c("A", "B", "C")
tm <- populate_transition_matrix(3, tmat, c("tpAtoA", "tpAtoB", "tpAtoC", "tpBtoB", "tpBtoC", "tpCtoC"),
table_param.df

Description
Parameter table created

Usage
table_param.df

Format
A 11 by 2 dataframe

Source
created on Jan 15, 2020
**trace_data.df**

**Trace matrix**

**Description**
Trace matrix

**Usage**
trace_data.df

**Format**
A 11 by 2 dataframe

**Source**
created on Nov 26, 2019 from tmat <- rbind(c(1, 2), c(3, 4)) colnames(tmat) <- rownames(tmat) <- c("Healthy", "Dead") tm <- transition_matrix(2, tmat, c(0.5, 0.5, 0, 1)) a <- health_state("Healthy", 1, TRUE) b <- health_state("Dead", 1, 0, TRUE) health_states <- combine_state(a, b) this.strategy <- strategy(tm, health_states, "intervention")

**transition_cost_util**  
Create the the values of cost and utility while transition

**Description**
Create the the values of cost and utility while transition

**Usage**
transition_cost_util(
    no_states,
    tmat_cost_util,
    list_values,
    name_states = NULL
)

**Arguments**

- no_states: number of the health states
- tmat_cost_util: A transition matrix for the cost/utility values in the format from the package 'mstate' use NA to indicate if the value is zero
- list_values: list of probabilities as in the order of transitions (row wise)
- name_states: names of the health states
Details

Similar to transition matrix but for denoting one time change during transitions

Value

value of the transition matrix

Examples

tmat_cost <- rbind(c(NA, 1), c(NA, NA))
colnames(tmat_cost) <- rownames(tmat_cost) <- c("Healthy", "Dead")
transition_cost_util(2, tmat_cost, list_values = c(500))

---

trial_data.df Example trial data

---

Description

Example trial data

Usage

trial_data.df

Format

A 31 by 33 dataframe

Source

created on Jan 15, 2020

---

use_coxph_survival

Get the parameter values using the survival analysis using cox proportional hazard

Description

Get the parameter values using the survival analysis using cox proportional hazard
**Usage**

```r
use_coxph_survival(
    param_to_be_estimated,
    dataset,
    indep_var,
    covariates,
    timevar_survival
)
```

**Arguments**

- `param_to_be_estimated`: parameter of interest
- `dataset`: data set to be provided
- `indep_var`: the independent variable (column name in data file)
- `covariates`: list of covariates - calculations to be done before passing
- `timevar_survival`: time variable for survival analysis, default is NA false by default

**Details**

plots baseline cumulative hazard function, survival function for each covariate while keeping the other fixed at the mean value (using plot_survival_cox_covariates), survival function for each combination of covariate using survfit (using plot_return_survival_curve) and test for cox regression results It also returns risk relative to mean (predicted at mean value of each covariate) along with the fit results coefficients, SE of coefficients, summary, and analysis of deviance

**Value**

the results of the regression analysis

**Examples**

```r
data_for_survival <- survival::aml
surv_estimated <- use_coxph_survival("status", data_for_survival, "x",
    covariates = NA, "time")

data_for_survival <- survival::lung
surv_estimated <- use_coxph_survival("status", data_for_survival, "sex",
    covariates = c("ph.ecog"), "time")
```
use_fh2_survival

Get the parameter values using the survival analysis using FH2 method

Usage

use_fh2_survival(
  param_to_be_estimated,
  dataset,
  indep_var,
  covariates,
  timevar_survival
)

Arguments

param_to_be_estimated  parameter of interest
dataset  data set to be provided
indep_var  the independent variable (column name in data file)
covariates  list of covariates
timevar_survival  time variable for survival analysis, default is NA false by default

Details

This function is for survival analysis using FH2. This plots the cumulative survival function for each combination of covariate If the covariate is numeric, R takes it as different levels. The plot uses the returned list of survfit and extracts the time and the strata from summary of the fit (implemented in plot_return_survival_curve function)

Value

the results of the regression analysis

Examples

data_for_survival <- survival::aml
surv_estimated <- use_fh2_survival("status", data_for_survival, "x",
covariates = NA, "time")
use_fh_survival

Get the parameter values using the survival analysis method FH

Description

Get the parameter values using the survival analysis method FH

Usage

use_fh_survival(
  param_to_be_estimated,
  dataset,
  indep_var,
  covariates,
  timevar_survival
)

Arguments

  param_to_be_estimated  parameter of interest
  dataset                data set to be provided
  indep_var              the independent variable (column name in data file)
  covariates             list of covariates
  timevar_survival       time variable for survival analysis, default is NA

Details

This function is for survival analysis using FH. This plots the cumulative survival function for each combination of covariate. If the covariate is numeric, R takes it as different levels. The plot uses the returned list of survfit and extracts the time and the strata from summary of the fit (implemented in plot_return_survival_curve function)

Value

the results of the regression analysis

Examples

data_for_survival <- survival::aml
surv_estimated <- use_fh_survival("status", data_for_survival, "x",
  covariates = NA, "time"
)
use_generalised_linear_mixed_model

*Function for generalised linear mixed model*

### Description

Function for generalised linear mixed model

### Usage

```r
use_generalised_linear_mixed_model(
    param_to_be_estimated,
    dataset,
    fix_eff,
    fix_eff_interact_vars,
    random_intercept_vars,
    nested_intercept_vars_pairs,
    cross_intercept_vars_pairs,
    uncorrel_slope_intercept_pairs,
    random_slope_intercept_pairs,
    family,
    link,
    package_mixed_model
)
```

### Arguments

- `param_to_be_estimated`: column name of dependent variable
- `dataset`: a dataframe
- `fix_eff`: names of variables as fixed effect predictors
- `fix_eff_interact_vars`: those of the fixed effect predictors that show interaction
- `random_intercept_vars`: names of variables for random intercept
- `nested_intercept_vars_pairs`: those of the random intercept variables with nested effect
- `cross_intercept_vars_pairs`: those of the random intercept variables with crossed effect
- `uncorrel_slope_intercept_pairs`: variables with no correlated intercepts
- `random_slope_intercept_pairs`: random slopes intercept pairs - this is a list of paired variables
- `family`: family of distributions for the response variable
- `link`: link function for the variances
use_generalised_linear_model

package_mixed_model
package to be used for mixed model

Value

result regression result with plot if success and -1, if failure

Examples

datafile <- system.file("extdata", "culcita_data.csv",
package = "packDAMipd")
dataset <- read.csv(datafile)
results1 = use_generalised_linear_mixed_model("predation",
dataset = datafile, fix_eff = c("ttt"), family = "binomial",
fix_eff_interact_vars = NULL, random_intercept_vars = c("block"),
nested_intercept_vars_pairs = NULL, cross_intercept_vars_pairs = NULL,
uncorrel_slope_intercept_pairs = NULL, random_slope_intercept_pairs = NULL,
link = NA, package_mixed_model = NA)

use_generalised_linear_model

############################################################################
Get the parameter values using logistic regression

Description

############################################################################ Get
the parameter values using logistic regression

Usage

use_generalised_linear_model(
  param_to_be_estimated,
  dataset,
  indep_var,
  family,
  covariates,
  interaction,
  naaction,
  link = NA
)
use_km_survival

Arguments

param_to_be_estimated
  parameter of interest
dataset
data set to be provided
indep_var
  the independent variable (column name in data file)
family
distribution name eg. for logistic regression - binomial
covariates
  list of covariates - calculations to be done before passing
interaction
  boolean value to indicate interaction in the case of linear regression
naaction
  action to be taken with the missing values
link
  link function if not the default for each family

details

This function returns the results and plots after doing linear regression Requires param to be estimated, dataset, independent variables and information on covariates, and interaction variables if there are. Uses form_expression_glm to create the expression as per R standard for e.g glm(y ~ x).
Returns the fit result, s summary results as returned by summary(), confidence interval for fit coefficients (ci_coeff), variance covariance matrix, cholesky decomposition matrix, results from correlation test, plot of diagnostic tests and model fit assumptions, plot of model prediction diagnostic include AIC, R2, and BIC. The results of the prediction ie predicted values for fixed other variables will be returned in prediction matrix

value

the results of the regression analysis

examples

```r
gm_result <- use_generalised_linear_model(
  param_to_be_estimated = "Direction",
  dataset = ISLR::Smarket, indep_var = "Lag1", family = "binomial",
  covariates = c("Lag2", "Lag3"),
  interaction = FALSE, naaction = "na.omit", link = NA)
```

use_km_survival

Get the parameter values using the Kaplan-Meier survival analysis

description

Get the parameter values using the Kaplan-Meier survival analysis
use_km_survival

Usage

use_km_survival(
  param_to_be_estimated,
  dataset,
  indep_var,
  covariates,
  timevar_survival
)

Arguments

param_to_be_estimated
  parameter of interest
dataset
  data set to be provided
indep_var
  the independent variable (column name in data file)
covariates
  list of covariates
timevar_survival
  time variable for survival analysis, default is NA

Details

This function is for survival analysis using Kaplan Meier. This plots the cumulative survival function for each combination of covariate. If the covariate is numeric, R takes it as different levels. The plot uses the returned list of survfit and extracts the time and the strata from summary of the fit (implemented in plot_return_survival_curve function)

Value

the results of the regression analysis, fit results, summary and plot

Examples

data_for_survival <- survival::aml
surv_estimated <- use_km_survival("status", data_for_survival, "x",
covariates = NA, "time")

data_for_survival <- survival::lung
surv_estimated <- use_km_survival("status", data_for_survival, "sex",
covariates = c("ph.ecog"), "time")
use_linear_mixed_model

Function for mixed effect regression

Description

Function for mixed effect regression

Usage

use_linear_mixed_model(
  param_to_be_estimated,
  dataset,
  fix_eff,
  fix_eff_interact_vars,
  random_intercept_vars,
  nested_intercept_vars_pairs,
  cross_intercept_vars_pairs,
  uncorrel_slope_intercept_pairs,
  random_slope_intercept_pairs,
  package_mixed_model
)

Arguments

param_to_be_estimated  column name of dependent variable
dataset  a dataframe
fix_eff  names of variables as fixed effect predictors
fix_eff_interact_vars,  those of the fixed effect predictors that show interaction
random_intercept_vars,  names of variables for random intercept
nested_intercept_vars_pairs,  those of the random intercept variables with nested effect
cross_intercept_vars_pairs,  those of the random intercept variables with crossed effect
uncorrel_slope_intercept_pairs,  variables with no correlated intercepts
random_slope_intercept_pairs,  random slopes intercept pairs - this is a list of paired variables
package_mixed_model  package to be used for mixed model
Value

result regression result with plot if success and -1, if failure

Examples

```r
datafile <- system.file("extdata", "data_linear_mixed_model.csv", package = "packDAMipd")
dataset = utils::read.table(datafile, header = TRUE, sep = ",", na.strings = "NA", dec = ",", strip.white = TRUE)
result <- use_linear_mixed_model("extro",
dataset = dataset,
fix_eff = c("open", "agree", "social"), fix_eff_interact_vars = NULL,
random_intercept_vars = c("school", "class"),
nested_intercept_vars_pairs = list(c("school", "class")),
cross_intercept_vars_pairs = NULL, uncorrel_slope_intercept_pairs = NULL,
random_slope_intercept_pairs = NULL, package_mixed_model = NA)
```

use_linear_regression

###########################################################################
Get the parameter values using the linear regression

Description

########################################################################### Get the parameter values using the linear regression

Usage

```r
use_linear_regression(
  param_to_be_estimated,
  dataset,
  indep_var,
  covariates,
  interaction
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>param_to_be_estimated</td>
<td>parameter of interest</td>
</tr>
<tr>
<td>dataset</td>
<td>data set to be provided</td>
</tr>
<tr>
<td>indep_var</td>
<td>the independent variable (column name in data file)</td>
</tr>
<tr>
<td>covariates</td>
<td>list of covariates-calculations to be done before passing</td>
</tr>
<tr>
<td>interaction</td>
<td>boolean value to indicate interaction in the case of linear regression, false by default</td>
</tr>
</tbody>
</table>

```r
```
**Details**

This function returns the results and plots after doing linear regression. Requires param to be estimated, dataset, independent variables and information on covariates, and interaction variables if there are. Uses form_expression_lm to create the expression as per R standard for e.g `lm(y ~ x)`. Returns the fit result's summary results as returned by summary(), confidence interval for fit coefficients (ci_coef), variance covariance matrix, cholesky decomposition matrix. Results from correlation test, plot of diagnostic tests and model fit assumptions, plot of model prediction diagnostic include AIC, R2, and BIC. The results of the prediction ie predicted values when each of covariate is fixed will be returned in prediction matrix predicted values will provide the mean value of param_to_to_estimated as calculated by the linear regression formula.

Ref: https://www.statmethods.net/stats/regression.html

**Value**

the results of the regression analysis

**Examples**

```r
results_lm <- use_linear_regression("dist",
    dataset = cars,
    indep_var = "speed", covariates = NA, interaction = FALSE)
```

```r
library(car)
results_lm <- use_linear_regression("mpg",
    dataset = mtcars,
    indep_var = "disp", covariates = c("hp", "wt", "drat"),
    interaction = FALSE)
```

---

**use_parametric_survival**

# Get the parameter values using the survival analysis parametric survival

**Description**

# Get the parameter values using the survival analysis parametric survival

**Usage**

```r
use_parametric_survival(
    param_to_be_estimated,
    dataset,
    indep_var,
)```
Arguments

- **param_to_be_estimated**: parameter of interest
- **dataset**: data set to be provided
- **indep_var**: the independent variable (column name in data file)
- **info_distribution**: distribution name eg. for logistic regression - binomial
- **covariates**: list of covariates
- **timevar_survival**: time variable for survival analysis, default is NA
- **cluster_var**: cluster variable for survival analysis

Details

This function is the last in the layer of function for parametric survival analysis. This then returns the parameters of interest, plots the results etc if the distribution is weibull it uses the package SurvRegCensCov for easy interpretation of results Returns the fit result, summary of regression, variance-covariance matrix of coeff, cholesky decomposition, the parameters that define the assumed distribution and the plot of model prediction Using survfit from survival package to plot the survival curve R’s weibull distribution is defined as std weibull in terms of a and b as \((a/b) \cdot (x/b)^{(a-1)} \cdot \exp((-x/b)^a)\) where a is the shape and b is the scale In HE the weibull distribution is parameterised as bit different it is like gamma.lambda. \(t^\gamma \cdot \exp(-\lambda t^\gamma)\) where gamma is the shape and lambda is the scale. The relationship is as below. HE's shape = rweibull_shape HE's scale = rweibull_scale \(^\gamma(-rweibull_shape)\) The survreg shape and scale are again bit different and they are rweibull’s shape and scale as below. rweibull_shape = 1/fit$scale rweibull_scale = exp(fit intercept) = exp(fit$coefficients) remember to use 1st of coefficients This has been utilised in SurvRegCensCov::ConvertWeibull predict() for survreg object with type = quantile will provide the failure times as survival function is 1-CDF of failure time.

Value

the results of the regression analysis

Examples

data_for_survival <- survival::lung
surv_estimated <- use_parametric_survival("status", 
data_for_survival, "sex", info_distribution = "weibull", 
covariates = c("ph.ecog"), "time")
use_seemingly_unrelated_regression

Bivariate regression for correlated observations

Description

Bivariate regression for correlated observations

Usage

use_seemingly_unrelated_regression(
  param1_to_be_estimated,
  param2_to_be_estimated,
  dataset,
  indep_var,
  covariates1,
  covariates2,
  interaction1,
  interaction2
)

Arguments

param1_to_be_estimated
  parameter of interest
param2_to_be_estimated
  parameter of interest
dataset
  data set to be provided
indep_var
  the independent variable (column name in data file)
covariates1
  list of covariates - for equation 1
covariates2
  list of covariates - for equation 2
interaction1
  boolean value to indicate interaction - for equation 1
interaction2
  boolean value to indicate interaction - for equation 2 false by default

Value

the results of the regression analysis

Examples

datafile <- system.file("extdata", "sureg_data.csv", package = "packDAMipd")
dataset <- read.csv(datafile, stringsAsFactors = TRUE)
results_sureg <- use_seemingly_unrelated_regression("read", "math",
  dataset = dataset,
  indep_var = "female", covariates1 = c("as.numeric(ses)", "socst"),
covariates = c("as.numeric(ses)", "science")
# Example usage

use_survival_analysis(covariates = c("as.numeric(ses)", "science"), interaction1 = FALSE, interaction2 = FALSE)

---

**Description**

Get the parameter values using the survival analysis

**Usage**

```r
use_survival_analysis(
  param_to_be_estimated, dataset, indep_var, info_get_method, info_distribution, covariates, timevar_survival, cluster_var = NA)
```

**Arguments**

- `param_to_be_estimated`: parameter of interest
- `dataset`: data set to be provided
- `indep_var`: the independent variable (column name in data file)
- `info_get_method`: additional information on methods e.g. Kaplan-Meier or hazard
- `info_distribution`: distribution name eg. for logistic regression - binomial
- `covariates`: list of covariates - calculations to be done before passing
- `timevar_survival`: time variable for survival analysis, default is NA
- `cluster_var`: cluster variable for survival analysis

**Details**

This function helps to get the parameter values after the survival analysis. Takes into account many different methods like KM, FH, Cox proportional etc. and then calls appropriate functions to do the survival analysis.
Value

the results of the regression analysis

Examples

data_for_survival <- survival::aml
surv_estimated_aml <- use_survival_analysis("status", data_for_survival, "x",
  info_get_method = "parametric", info_distribution = "weibull",
  covariates = NA, "time")

utility_data.df       utility matrix

Description

utility matrix

Usage

utility_data.df

Format

A 11 by 2 dataframe

Source

created on Nov 26, 2019 from tmat <- rbind(c(1, 2), c(3, 4)) colnames(tmat) <- rownames(tmat) <-
c("Healthy", "Dead") tm <- transition_matrix(2, tmat, c(0.5, 0.5, 0, 1)) a <- health_state("Healthy",
1, 1, FALSE) b <- health_state("Dead", 1, 0, TRUE) health_states <- combine_state(a, b) this.strategy <- strategy(tm, health_states, "intervention")

value_ADL_scores_IPD  Function to convert ADL scores to a T score

Description

Function to convert ADL scores to a T score
value_eq5d3L_IPD

Usage

\texttt{value\_ADL\_scores\_IPD(}
\texttt{  ind\_part\_data,}
\texttt{  adl\_related\_words,}
\texttt{  adl\_ncode,}
\texttt{  adl\_scoring\_table = NULL}
\texttt{)}

Arguments

\texttt{ind\_part\_data} a data frame containing IPD data
\texttt{adl\_related\_words} related words to find out which columns contain adl data
\texttt{adl\_ncode} non response code for ADL
\texttt{adl\_scoring\_table} ADL scoring table, if given as NULL use the default one

Value

ADL scores converted to T score included modified data, if success \(-1\), if failure

Examples

\begin{verbatim}
  datafile <- system.file("extdata", "trial_data.csv", package = "packDAMipd")
  trial_data <- load_trial_data(datafile)
  value_ADL_scores_IPD(trial_data,c("tpi"), NA, adl_scoring_table = NULL)
\end{verbatim}

value_eq5d3L_IPD

\textit{Function to add EQ5D3L scores to IPD data}

Description

Function to add EQ5D3L scores to IPD data

Usage

\texttt{value\_eq5d3L\_IPD(ind\_part\_data, eq5d\_ncode)}

Arguments

\texttt{ind\_part\_data} a dataframe
\texttt{eq5d\_ncode} non response code for EQ5D3L, default is NA

Value

qaly included modified data, if success \(-1\), if failure
value_eq5d5L_IPD

Source
http://eprints.whiterose.ac.uk/121473/1/Devlin_et_al-2017-Health_Economics.pdf

Examples

datafile <- system.file("extdata", "trial_data.csv", package = "packDAMipd")
trial_data <- load_trial_data(datafile)
value_eq5d5L_IPD(trial_data, NA)

value_eq5d5L_IPD  Function to add EQ5D5L scores to IPD data

Description
Function to add EQ5D5L scores to IPD data

Usage

value_eq5d5L_IPD(ind_part_data, eq5d_nrcode)

Arguments

ind_part_data  a dataframe
eq5d_nrcode  non response code for EQ5D5L, default is NA

Value
qaly included modified data, if success -1, if failure

Source
http://eprints.whiterose.ac.uk/121473/1/Devlin_et_al-2017-Health_Economics.pdf

Examples

datafile <- system.file("extdata", "trial_data.csv", package = "packDAMipd")
trial_data <- load_trial_data(datafile)
value_eq5d5L_IPD(trial_data, NA)
value_promis3a_scores_IPD

*Function to convert promis3a scores to a T score*

**Description**

Function to convert promis3a scores to a T score

**Usage**

```r
value_promis3a_scores_IPD(
  ind_part_data,
  promis3a_related_words,
  promis3a_nrcode,
  promis3a_scoring_table = NULL
)
```

**Arguments**

- `ind_part_data`  
  a data frame containing IPD data
- `promis3a_related_words`  
  related words to find out which columns contain promis3a data
- `promis3a_nrcode`  
  non response code for promis3a
- `promis3a_scoring_table`  
  promis3a scoring table, if given as NULL use the default one

**Value**

promis3a scores converted to T score included modified data, if success -1, if failure

**Examples**

```r
trial_data <- data.frame("tpi.q1" = c(1, 2),
"tpi.q2" = c(1, 2), "tpi.q3" = c(1, 2))
results <- value_promis3a_scores_IPD(trial_data, c("tpi"), NA, NULL)
```
value_Shows_IPD  
*Function to estimate the cost of tablets taken (from IPD)*

**Description**
Function to estimate the cost of tablets taken (from IPD)

**Usage**

```
value_Shows_IPD(ind_part_data, shows_related_words, shows_nrcode)
```

**Arguments**

- `ind_part_data`: a data frame containing IPD
- `shows_related_words`: a data frame containing IPD
- `shows_nrcode`: non response code for ADL, default is NA

**Value**
sum of scores, if success -1, if failure

**Examples**
```
datafile <- system.file("extdata", "trial_data.csv", package = "packDAMipd")
trial_data <- load_trial_data(datafile)
value_Shows_IPD(trial_data, "qsy", NA)
```

---

word2num  
*Function to check the variable null or NA*

**Description**
Function to check the variable null or NA

**Usage**

```
word2num(word)
```

**Arguments**

- `word`: word for the number

**Details**
https://stackoverflow.com/questions/18332463/convert-written-number-to-number-in-r
Value

return the number

Examples

answer <- word2num("one forty one")
answer <- word2num("forty one and five hundred")
answer <- word2num("five thousand two hundred and eight")
Index

* datasets
  adl_scoring.df, 6
  blank.df, 7
  cost_data.df, 41
  promis3a_scoring.df, 121
  table_param.df, 128
  trace_data.df, 129
  trial_data.df, 130
  utility_data.df, 144

  add_entries_sameuse_timepoint, 5
  adl_scoring.df, 6
  assign_parameters, 6

  blank.df, 7

  calculate_icer_nmb, 8
  check_equal_columncontents_NAomitted, 11
  check_equal_sumcolumncontents_NAomitted, 12
  check_link_glm, 13
  check_list_markov_models, 13
  check_null_na, 14
  check_trans_prob, 15
  check_treatment_arm, 16
  check_values_states, 16
  checks_markov_pick_method, 9
  checks_plot_dsa, 10
  combine_markov, 17
  combine_state, 18
  convert_freq_diff_basis, 19
  convert_to_given_timeperiod, 19
  convert_volume_basis, 20
  convert_weight_diff_basis, 20
  convert_wtpertimediff_basis, 21
  convert_wtpervoldiff_basis, 22
  cost_data.df, 41
  costing_AandE_admission, 22
  costing_inpatient_daycase_admission, 24
  costing_opioid_liquids_averageMED_long, 25
  costing_opioid_liquids_averageMED_wide, 27
  costing_opioid_patches_averageMED_long, 29
  costing_opioid_patches_averageMED_wide, 31
  costing_opioid_tables_averageMED_long, 33
  costing_opioid_tables_averageMED_wide, 35
  costing_opioid_tables_MED_wide, 37
  costing_resource_use, 39
  create_new_dataset, 42
  create_shorttable_from_gtsummary_compare_twogroups_timpoints, 42
  create_table_from_gtsummary_compare_twogroups, 43

  define_parameters, 44
  define_parameters_psa, 45
  define_parameters_sens_anal, 46
  define_transition_table, 47
  do_psa, 48
  do_sensitivity_analysis, 49

  encode_codes_data, 50
  eval_assign_trans_prob, 51
  eval_assign_values_states, 52

  find_glm_distribution, 53
  find_keyword_rand_generation, 54
  find_keyword_regression_method, 54
  find_parameters_btn_operators, 55
  find_required_parameter_combs, 56
  find_rowwise_sum_multiplecol, 56
  find_survreg_distribution, 57
INDEX

form_expression_glm, 58
form_expression_lm, 59
form_expression_mixed_model_lme4, 59

generate_wt_time_units, 61
generate_wt_vol_units, 61
get_age_details, 62
get_colnames_codedvalues, 63
get_cost_AandE_code, 63
get_cost_AandE_description, 64
get_cost_ip_dc_description, 65
generate_wt_vol_units, 61
get_doses_combination, 67
generate_wt_time_units, 61
get_doses_combination_units, 68
get_eq5d_details, 68
get_extension_file, 69
get_gender_details, 70
get_mean_sd_age, 70
get_slope_intercept, 78
get_slope_intercept_cross, 79
get_slope_intercept_nested, 80
get_single_col_multiple_pattern, 78
get_var_state, 82
health_state, 82
init_trace, 83
init_trace_sjtime, 84
keep_results_plot_dsa, 84
list_paramwise_psa_result, 85
load_trial_data, 86
map_eq5d5Lto3L_VanHout, 87
map_resource_use_categories, 88
markov_model, 89
markov_model_sojourntime, 90
microcosting_liquids_long, 92
microcosting_liquids_wide, 94
microcosting_patches_long, 97
microcosting_patches_wide, 99
microcosting_tablets_long, 101
microcosting_tablets_wide, 103
plot_ceac_psa, 105
plot_dsa, 105
plot_dsa_difference, 107
plot_dsa_icer_range, 108
plot_dsa_nmb_range, 109
plot_dsa_others_range, 109
plot_efficiency_frontier, 110
plot_histogram_onetimepoint_twogroups, 110
plot_meanSE_longitudinal_twogroups, 111
plot_model, 112
plot_nmb_lambda, 113
plot_prediction_parametric_survival, 114
plot_return_residual_cox, 115
plot_return_residual_survival, 116
plot_return_survival_curve, 117
plot_transition_matrix, 118
promis3a_scoring.df, 121
report_sensitivity_analysis, 121
return0_if_not_null_na, 122
return_equal_liststring_col, 123
return_equal_liststring_listcol, 124
return_equal_str_col, 125
set_var_state, 125
strategy, 126
summary_plot_psa, 127

table_param.df, 128
trial_data.df, 129
transition_cost_util, 129

use_coxph_survival, 130
use_fh2_survival, 132
use_fh_survival, 133
use_generalised_linear_mixed_model, 134
use_generalised_linear_model, 135
use_km_survival, 136
use_linear_mixed_model, 138
use_linear_regression, 139
use_parametric_survival, 140
use_seemingly_unrelated_regression, 142
use_survival_analysis, 143
utility_data.df, 144

value_ADL_scores_IPD, 144
value_eq5d3L_IPD, 145
value_eq5d5L_IPD, 146
value_promis3a_scores_IPD, 147
value_Shows_IPD, 148

word2num, 148