Package ‘INSPECTumours’

October 12, 2022

Title IN-vivo reSPonsE Classification of Tumours
Version 0.1.0
Maintainer Bairu Zhang <bairu.zhang@astrazeneca.com>
Description This is a shiny app used for the statistical classifying and analysing pre-clinical tumour responses.
License Apache License (== 2)
Depends R (>= 3.5.0)
Imports brms, dplyr, DT, ggeffects, ggplot2, knitr, lme4, modelr, pander, plotly, purrr, readxl, rlang, rmarkdown, shiny, shinyalert, shinyFeedback, shinyjs, shinytoastr, shinyvalidate, tidybayes, tidyrr, vroom, waiter
Suggests spelling, lintr, testthat
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Language en-US
NeedsCompilation no
Author Bairu Zhang [cre, aut], Olga Muraeva [aut], Natasha Karp [aut]
Repository CRAN
Date/Publication 2022-05-06 12:10:02 UTC

R topics documented:

  aggregate_study_info .................................................. 2
  animal_info_classification ............................................. 3
  assess_efficacy .......................................................... 3
  below_min_points ....................................................... 4
  calc_gr ................................................................. 4
  calc_probability ........................................................ 5
aggregate_study_info

create a table with aggregated data: each row contains information about control and treatments of a single study

Usage

aggregate_study_info(df)
animal_info_classification

Arguments

   df  data.frame

Value

   data.frame

---

animal_info_classification

   Generate table representing number of animals in classification groups

Description

   Generate table representing number of animals in classification groups

Usage

   animal_info_classification(data)

Arguments

   data  final classification data

Value

   data frame

---

assess_efficacy

   Credible interval (or say “Bayesian confidence interval”) of the mean difference between two groups (treatment and reference) is used to assess the efficacy. If 0 falls outside the interval, the drug was considered significantly effective

Description

   Credible interval (or say “Bayesian confidence interval”) of the mean difference between two groups (treatment and reference) is used to assess the efficacy. If 0 falls outside the interval, the drug was considered significantly effective

Usage

   assess_efficacy(data, reference = "Control")
below_min_points  

makes df with data to be excluded

Usage

below_min_points(df, min_points)

Arguments

df  initial data frame

min_points  minimum number of data points for one animal_id per study

Value

df

calc_gr  

Function to return rate of growth (e.g. the slope after a log transformation of the tumour data against time)

Description

Function to return rate of growth (e.g. the slope after a log transformation of the tumour data against time)

Usage

calc_gr(df, log_tv = "log_tv", day = "day")

Arguments

df  subset, one animal_id

log_tv  name of the column, tumour volume

day  name of the column, days

Arguments

data  prediction results

reference  name of the reference treatment

Value

dataframe with information about drug efficacy
**calc_probability**

**Value**

tibble with GR and GR_SE

---

**Description**

Calculate probability of categories

**Usage**

calc_probability(data)

**Arguments**

data data frame with predictions

**Value**

data frame

---

**calc_survived**

*Calculate percentage of survived animals*

---

**Description**

Calculate percentage of survived animals

**Usage**

calc_survived(df)

**Arguments**

df data frame

**Value**

data frame
**change_time_multi**  
*Get an array with change_time for studies from the population-level effects, multiple studies*

**Description**
Get an array with change_time for studies from the population-level effects, multiple studies

**Usage**
```r
change_time_multi(model)
```

**Arguments**
- `model` an object of class `brmsfit`

**Value**
data frame

**change_time_single**  
*Get a change time from the population-level effects, single study*

**Description**
Get a change time from the population-level effects, single study

**Usage**
```r
change_time_single(model)
```

**Arguments**
- `model` an object of class `brmsfit`

**Value**
a numeric vector of length one
**classify_data_point**  
*Classify individual data points as Responders or Non-responders*

**Description**  
Classify individual data points as Responders or Non-responders

**Usage**  
```r  
classify_data_point(df_newstudy, pred_newstudy)  
```

**Arguments**  
- `df_newstudy`: data from new study  
- `pred_newstudy`: data frame with predictions

**Value**  
data frame with "Responder"/"Non-responder" for individual data points

**classify_subcategories**  
*Make predictions for subcategories*

**Description**  
Make predictions for subcategories

**Usage**  
```r  
classify_subcategories(data, model)  
```

**Arguments**  
- `data`: data frame with classification results  
- `model`: object of class brmsfit

**Value**  
data frame
classify_type_responder

Classify tumour based on the growth rate and the p_value for a two-sided T test. Tumour will be considered as "Non-responder", "Modest responder", "Stable responder" or "Regressing responder".

Description

Classify tumour based on the growth rate and the p_value for a two-sided T test. Tumour will be considered as "Non-responder", "Modest responder", "Stable responder" or "Regressing responder".

Usage

classify_type_responder(df)

Arguments

df: data frame

Value

data frame with a new column classify_tumour

clean_string

function to remove hyphens, underscores, spaces and transform to lowercase

Description

function to remove hyphens, underscores, spaces and transform to lowercase

Usage

clean_string(string)

Arguments

string: to modify

Value

modified string
control_growth_plot  

Function to plot a control growth profile

Description
Function to plot a control growth profile

Usage
control_growth_plot(df, model_type, col_palette)

Arguments
- df: data frame
- model_type: string
- col_palette: character palette

Value
ggplot object

example_data  
Tumour volume data over time for in-vivo studies

Description
A dataset containing the repeatedly measurements of tumour volume data over time for individual animals.

Usage
example_data

Format
A data frame with 1462 rows and 6 variables:
- study: study identifier
- group: group identifier
- treatment: treatment type
- animal_id: animal identifier
- day: day after implant
- tumour_volume: volume in mm3
### exclude_data

*Filter rows to exclude from the analysis*

**Description**

Filter rows to exclude from the analysis

**Usage**

```r
exclude_data(df, study_id_ex, animal_id_ex, day_ex, reason)
```

**Arguments**

- `df`: initial df
- `study_id_ex`: string: study id
- `animal_id_ex`: string: animal id
- `day_ex`: string: day
- `reason`: string: why it should be excluded

**Value**

dataframe with rows that meets exclusion criteria

---

### expand_palette

*Function to expand a vector of colors if needed*

**Description**

Function to expand a vector of colors if needed

**Usage**

```r
expand_palette(col_palette, n)
```

**Arguments**

- `col_palette`: character palette to color the treatments
- `n`: how many colors are needed

**Value**

a character vector of colors
**f_start**

*Calculate coefficients for a nonlinear model*

**Description**

Calculate coefficients for a nonlinear model

**Usage**

```r
f_start(df, x, y, r_change)
```

**Arguments**

- `df`: data frame with `x` as a predictor and `y` is an outcome
- `x`: predictor string
- `y`: outcome string
- `r_change`: numeric

**Value**

list of coefficients

---

**get_responder**

*Classify tumour based on response status of individuals*

**Description**

Classify tumour based on response status of individuals

**Usage**

```r
get_responder(x, n)
```

**Arguments**

- `x`: character vector with response statuses of one animal
- `n`: consecutive measurements for classification

**Value**

"Responder" or "Non-responder"
guess_match  
function to search for the possible critical columns in a data.frame

Description
function to search for the possible critical columns in a data.frame

Usage
guess_match(colnames_df, crit_cols)

Arguments
colnames_df a character vector with names
crit_cols  a character vector

Value
list: possible match to each critical column

hide_outliers  Function to hide outliers in boxplots with jitterdodge as suggested

Description
Function to hide outliers in boxplots with jitterdodge as suggested

Usage
hide_outliers(x)

Arguments
x  plotly object

Value
plotly object without boxplot outliers
load_data

function to read data from users (.csv or .xlsx files)

Description
function to read data from users (.csv or .xlsx files)

Usage
load_data(path, name)

Arguments
- path: path to a temp file
- name: filename provided by the web browser

Value
data frame

make_terms
Create a character vector with the names of terms from model, for which predictions should be displayed Specific values are specified in square brackets

Description
Create a character vector with the names of terms from model, for which predictions should be displayed Specific values are specified in square brackets

Usage
make_terms(days, studies = NULL)

Arguments
- days: vector with days with which to predict
- studies: vector with studies with which to predict

Value
vector with values for predictions
**model_control**

*Build model and make predictions*

**Description**

Build model and make predictions

**Usage**

`model_control(df_control, df_newstudy, method, end_day)`

**Arguments**

- `df_control`: data frame with control data (including historical control, if provided)
- `df_newstudy`: data frame, data from new study
- `method`: "Two-stage non-linear model" or "Linear model"
- `end_day`: period of time used for the statistical modelling of the control data

**Value**

- list: two data frames with prediction results (for new study and for control data)

---

**notify_error_and_reset_input**

*Display a popup message and reset fileInput*

**Description**

Display a popup message and reset fileInput

**Usage**

`notify_error_and_reset_input(message_text)`

**Arguments**

- `message_text`: the modal’s text
**ordered_regression**

*Fit model (Bayesian ordered logistic regression)*

**Description**

Fit model (Bayesian ordered logistic regression)

**Usage**

`ordered_regression(df, formula, n_cores)`

**Arguments**

- `df`: data frame with classification results. Tumour classification is converted into ordinal data
- `formula`: string
- `n_cores`: number of cores to use

**Value**

object of class brmsfit

---

**plotly_volume**

*Create volume plot for one-batch data*

**Description**

Create volume plot for one-batch data

**Usage**

```r
plotly_volume(
    df, 
    col_palette = NULL, 
    faceting_var, 
    y_name, 
    y_var, 
    p_title, 
    ... 
)
```
Arguments

df       data.frame, single-batch long format
col_palette  character palette to color the treatments
faceting_var  string
y_name       string
y_var       string: column name for y axis
p_title       plot title
...                arguments passed to plot_ly

Value

plotly object

plot_animal_info  Plot representing number of animals in classification groups

Description

Plot representing number of animals in classification groups

Usage

plot_animal_info(data, col_palette)

Arguments

data       final classification data
col_palette  character palette

Value

ggplot object
plot_class_gr

Function to plot classification over growth rate

Description
Function to plot classification over growth rate

Usage
plot_class_gr(df, col_palette)

Arguments
- df: data frame
- col_palette: character palette

Value
ggplot object

plot_class_tv

Function to plot classification over tumour volume

Description
Function to plot classification over tumour volume

Usage
plot_class_tv(df, col_palette, title_name)

Arguments
- df: data frame
- col_palette: named vector
- title_name: character

Value
ggplot object
plot_proportions  

**Description**

Plot estimated proportions

**Usage**

```r
plot_proportions(data, col_palette)
```

**Arguments**

- `data`: table of the category prediction
- `col_palette`: character palette

plot_waterfall  

**Description**

Function to plot waterfall

**Usage**

```r
plot_waterfall(df, col_palette, study_name)
```

**Arguments**

- `df`: data frame
- `col_palette`: character palette
- `study_name`: string: to show on title

**Value**

`ggplot` object
predict_lm

Make predictions, linear model

Description

Make predictions, linear model

Usage

predict_lm(model, newdata, single)

Arguments

model a model object
newdata data frame in which to look for variables with which to predict
single logical: TRUE if single study experiment

Value

data frame with predictions

predict_nlm_multi

Make predictions based on non-linear model, multiple studies

Description

Make predictions based on non-linear model, multiple studies

Usage

predict_nlm_multi(model, newdata, change_time)

Arguments

model an object of class brmsfit
newdata data frame in which to look for variables with which to predict
change_time data frame

Value

data frame with predictions
**predict_nlm_single**  
Make predictions based on non-linear model, single study

**Description**
Make predictions based on non-linear model, single study

**Usage**
predict_nlm_single(model, newdata, change_time)

**Arguments**
- `model`: an object of class `brmsfit`
- `newdata`: data frame in which to look for variables with which to predict
- `change_time`: numeric

**Value**
data frame with predictions

**predict_regr_model**  
Make predictions

**Description**
Make predictions

**Usage**
predict_regr_model(model, df)

**Arguments**
- `model`: object of class `brmsfit`
- `df`: data frame with classification results

**Value**
data frame
**run_app**

*Run the Shiny Application*

**Description**

Run the Shiny Application

**Usage**

```r
run_app(...)```

**Arguments**

- `...`: additional options passed to `shinyApp()`

**Value**

No return value, called for the shiny app interface

---

**run_nl_model**

*Fit nonlinear model - continuous hinge function*

**Description**

Fit nonlinear model - continuous hinge function

**Usage**

```r
run_nl_model(start, df_mod, formula, n_cores)```

**Arguments**

- `start`: df with coefficients
- `df_mod`: data of all variables used in the model
- `formula`: an object of class `brmsformula`
- `n_cores`: number of cores to use

**Value**

object of class `brmsfit`
set_waiter

**Description**

Set up a waiting screen

**Usage**

```python
set_waiter(header)
```

**Arguments**

- `header`: text to display on loading screen

**Value**

object of a class waiter
Index

* datasets
  - example_data, 9

aggregate_study_info, 2
animal_info_classification, 3
assess_efficacy, 3
below_min_points, 4
calc_gr, 4
calc_probability, 5
calc_survived, 5
change_time_multi, 6
change_time_single, 6
classify_data_point, 7
classify_subcategories, 7
classify_type_responder, 8
clean_string, 8
control_growth_plot, 9

dataset

example_data, 9
exclude_data, 10
expand_palette, 10

f_start, 11
get_responder, 11
guess_match, 12

hide_outliers, 12

load_data, 13

make_terms, 13
model_control, 14

notify_error_and_reset_input, 14

ordered_regression, 15

plot_animal_info, 16
plot_class_gr, 17
plot_class_tv, 17
plot_proportions, 18
plot_waterfall, 18
plotly_volume, 15
predict_lm, 19
predict_nlm_multi, 19
predict_nlm_single, 20
predict_regr_model, 20

run_app, 21
run_nl_model, 21

set_waiter, 22