Package: hipecR (via r-universe)

May 22, 2026

Type Package

Title Tools for Analysing HIPEC Patient Data

Version 2.0.0

Maintainer Tarkan Jaeger <tarkan.jaeger@gmail.com>

Description Provides helper functions for analysing patient data in hyperthermic intraperitoneal chemotherapy (HIPEC) workflows. Includes functions to estimate peritoneal surface area (PSA), summarise registry data, and produce reporting graphics. Body surface area calculations are based on Du Bois and Du Bois (1916) <doi:10.1001/archinte.1916.00080130010002>.

License MIT + file LICENSE

URL https://github.com/TarJae/hipecR, https://tarjae.github.io/hipecR/

BugReports https://github.com/TarJae/hipecR/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Suggests clipr, DiagrammeR, gifski, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, tibble, shiny, shinyWidgets, formatR, dlookr, ggpubr, stringr, ggplot2, forcats, survminer, survival, labelled, gganimate, magick, tidyr, rlang, curl, htmltools, webshot2

Config/pak/sysreqs libabsl-dev chromium cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev make libharfbuzz-dev libmagick++-dev gsfonts libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

Repository https://tarjae.r-universe.dev

Date/Publication 2026-02-21 21:10:15 UTC

RemoteUrl https://github.com/tarjae/hipecr

RemoteRef HEAD

RemoteSha e5fd1f674090f374832c7621bc403a8f7b847df5

Contents

`tar_age` *Calculate person age*

Description

This function calculates the age of a person by given birthdate and the end date

Usage

```
tar_age(birth_date, x_date)
```

Arguments

birth_date	A date value, e.g. "1974-11-23".
x_date	A date value until the age lasts, e.g. "2023-02-16".

Value

A numeric value in (years)

Examples

```
tar_age("1974-11-23", "2023-02-16")
```

---

tar_animbar *Create Animated Bar Chart*

Description

This function creates an animated bar chart with customizable colors and orientation. The resulting GIF can be saved to a specified filename.

Usage

```
tar_animbar(
  x,
  filename = "my_animated_bar",
  color1 = "steelblue1",
  color2 = "purple3",
  title = "%",
  horizontal = FALSE,
  total = 100,
  fps = 24,
  base_duration = 15,
  output_dir = tempdir(),
  overwrite = FALSE,
  ask = interactive()
)
```

Arguments

x	An integer indicating the value for the bar chart.
filename	A string specifying the name of the output file (default: "my_animated_bar").
color1	A string specifying the color for the first part of the bar (default: "steelblue1").
color2	A string specifying the color for the second part of the bar (default: "purple3").
title	A string specifying the title suffix for the animation (default: percent sign).

horizontal	A logical value indicating whether the bar chart should be horizontal (default: FALSE).
total	An integer indicating the maximum value for the bar chart scale (default: 100).
fps	Frames per second for the animation (default: 24).
base_duration	Base duration (seconds) for a full-scale animation (default: 15).
output_dir	Output directory for the GIF (default: temporary directory).
overwrite	Logical; overwrite an existing GIF (default: FALSE).
ask	Logical; if TRUE, prompt when file exists (default: interactive()).

Details

This function generates an animated bar chart with customizable colors and orientation. The resulting GIF can be saved to a specified filename.

Value

Invisible path to the generated GIF file.

Examples

```
if (interactive() && requireNamespace("gifski", quietly = TRUE)) {
  tar_animbar(x = 10, filename = "a3", color1 = "gold", color2 = "purple3",
              horizontal = TRUE, total = 10, fps = 5, base_duration = 1,
              output_dir = tempdir(), overwrite = TRUE, ask = FALSE)
  tar_animbar(x = 5, filename = "a4", color1 = "steelblue", color2 = "purple3",
              horizontal = FALSE, total = 10, fps = 5, base_duration = 1,
              output_dir = tempdir(), overwrite = TRUE, ask = FALSE)
  tar_animbar(x = 3, filename = "rapido_LRR", horizontal = FALSE, total = 10,
              fps = 5, base_duration = 1, output_dir = tempdir(),
              overwrite = TRUE, ask = FALSE)
}
```

---

tar_bmi *Calculating BMI (kg/m^2)*

Description

Calculating BMI (kg/m^2)

Usage

```
tar_bmi(weight, height)
```

Arguments

weight	Numeric, kg
height	Numeric, cm

Value

body mass index BMI

Examples

```
tar_bmi(weight = 100, height = 190)
```

---

tar_bsa *Body surface area calculator in m^2 (DuBois and DuBois formula)*

Description

This function calculates the body surface area with the DuBois method. DuBois D. A formula to estimate the approximate surface area if height and body mass are known. Arch Intern Med 1916;17:863-71.

Usage

```
tar_bsa(height, weight)
```

Arguments

height	A numeric value in cm
weight	A numeric value in kg

Value

A numeric value in m^2

Examples

```
tar_bsa(height = 180, weight = 80) # ~1.996421 m^2
```

tar_cast_data *Cast registry data*

Description

Coerce raw input into a one-row data frame with registry fields.

Usage

```
tar_cast_data(data)
```

Arguments

data	A list or data frame with registry fields.

Value

A one-row data frame with standard column types.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_count_na *NA counter*

Description

Counts NA values across columns.

Usage

```
tar_count_na(df)
```

Arguments

df	A data frame or tibble.

Value

A named numeric vector with NA counts per column.

Examples

```
if(interactive()){
 tar_count_na(mtcars)
 }
```

`tar_count_top` *Count Top Categories in a Factor*

Description

This function transforms the specified factor variable within a dataframe to lump all but the top n most frequent levels into an 'Other' category and then computes the count of each level.

Usage

```
tar_count_top(df, var, n = 5)
```

Arguments

`df`	A data frame containing the variable to be manipulated.
`var`	The variable (unquoted) within the data frame.
`n`	The number of top levels to keep before lumping others into 'Other'; default is 5.

Details

This function leverages 'dplyr' for data manipulation and 'forcats' for managing factor levels. It is particularly useful in data summarization where the focus is on the most frequent categories.

Value

A dataframe showing the count of each level including 'Other' for all lumped lesser categories.

Examples

```
if(interactive()){
 data <- data.frame(color = c("red", "blue", "green", "blue", "blue", "red", "yellow", "red"))
  print(tar_count_top(data, color, n = 2))
 }
```

tar_create_data *Create registry record*

Description

Add a new record to the in-memory ‘responses‘ data frame.

Usage

```
tar_create_data(data)
```

Arguments

data	A named list or data frame with registry fields.

Value

Invisible NULL.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_create_default_record *Create default registry record*

Description

Return an empty default registry record.

Usage

```
tar_create_default_record()
```

Value

A one-row data frame with default values.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_dateInput *tar_dateinput*

Description

Handles date format in Austrian HIPEC Registry(c)

Usage

```
tar_dateInput(inputId, label)
```

Arguments

inputId	character
label	character

Details

Wraps shinyWidgets::airDatepickerInput with fixed defaults.

Value

A date input in dd-MM-yyyy format.

See Also

airDatepicker

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_delete_data *Delete registry record*

Description

Delete a record from ‘responses‘ by id.

Usage

```
tar_delete_data(data)
```

Arguments

data	A list or data frame containing an ‘id‘ field.

Value

Invisible NULL.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

---

tar_dgav_lookup *Look-up function for DGAV registry IDs/names*

Description

This function takes a data frame ‘df‘ and a vector of key values ‘mykeyvalues‘. It returns an unnamed vector of corresponding results by performing lookups in the data frame.

Usage

```
tar_dgav_lookup(df, mykeyvalues)
```

Arguments

df	data frame with Name and azl number
mykeyvalues	is a vector containing the key values to be looked up in the data frame. These key values can be either numeric or character strings. The function processes each key value individually to determine if it matches entries in the specified columns of the data frame

Value

a key or value, depending on what is wanted

Examples

```
# Example data
df <- data.frame(
  Name = c("Hans", "Maria", "Franz HUBER"),
  azl = c("1006341612", "1040405318", "1060707219"),
  stringsAsFactors = FALSE
)

tar_dgav_lookup(df, "HUBER")       # Should return "1060707219"
```

```
tar_dgav_lookup(df, "huber")       # Should also return "1060707219"
tar_dgav_lookup(df, "Franz")       # Should return "1060707219"
tar_dgav_lookup(df, 1006341612)  # Should return "Hans"
tar_dgav_lookup(df, c(1006341612, 1040405318)) # Should return "Hans"  "Maria"

# Function to handle inputs for data frame
```

---

tar_flag *Create a combined flag plot*

Description

This function downloads flag images of specified countries, combines them vertically with spaces in between, and saves the combined image to a file.

Usage

```
tar_flag(
  countries,
  name,
  output_dir = tempdir(),
  overwrite = FALSE,
  ask = interactive()
)
```

Arguments

countries	A character vector of country codes (ISO 3166\-1 alpha\-2).
name	A character string for the name of the study. This will be used to construct the output file name.
output_dir	A directory for the output file (default: temporary directory).
overwrite	Logical; overwrite an existing PNG (default: FALSE).
ask	Logical; if TRUE, prompt when file exists (default: interactive()).

Value

Invisible path to the saved PNG file.

Examples

```
if (interactive()) {
  tar_flag(c("nl", "se", "es", "si", "dk", "no", "us"), "rapido")
  tar_flag(c("at", "de"), "demo", output_dir = tempdir(), overwrite = TRUE)
}
```

tar_footnote *Add Footnote to Graph and Save as PNG*

Description

This function takes a ‘grViz‘ graph object, adds a footnote with specified style and text, and saves the graph along with the footnote as a PNG file. The PNG file is named based on the graph object name.

Usage

```
tar_footnote(
  graph,
  style,
  string,
  filename = NULL,
  output_dir = tempdir(),
  overwrite = FALSE,
  ask = interactive()
)
```

Arguments

graph	A ‘grViz‘ graph object.
style	A character string specifying the CSS style for the footnote.
string	A character string specifying the footnote text.
filename	Optional output filename (without extension). Defaults to the variable name of ‘graph‘ if possible.
output_dir	Output directory (default: temporary directory).
overwrite	Logical; overwrite an existing PNG (default: FALSE).
ask	Logical; if TRUE, prompt when file exists (default: interactive()).

Value

None. The function saves a PNG file in the working directory.

Examples

```
if (interactive() && requireNamespace("DiagrammeR", quietly = TRUE)) {
graph <- DiagrammeR::grViz("
digraph {
graph [layout = dot]
node [shape = box]
a [label = 'A']
b [label = 'B']
a -> {b}
```

```
}
")

style <- "color: grey; font-family: Arial; text-align: left; font-size: 10px;"
string <- paste(
  "CNCT = Consolidation Chemotherapy,",
  "CRT = Chemoradiotherapy,",
  "CT = Chemotherapy,",
  "INCT = Induction Chemotherapy,",
  "SCRT = Short-course Radiotherapy,",
  "TME = Total Mesorectal Excision,",
  "WW = Watch and Wait,",
  sep = " "
)

tar_footnote(graph, style, string)
}
```

`tar_get_next_id` *Get next registry ID*

Description

Get the next numeric ID based on the ‘responses‘ data frame.

Usage

```
tar_get_next_id()
```

Value

Integer ID.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_get_table_metadata *Registry field metadata*

Description

Create a named vector of fields and labels for the registry.

Usage

```
tar_get_table_metadata(variable = fieldsAll, label = label_names)
```

Arguments

variable	Character vector of field names. Default: fieldsAll.
label	Character vector of labels. Default: label_names.

Details

Expects ‘variable‘ and ‘label‘ to be the same length.

Value

A list with element ‘fields‘, a named character vector.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_ggplot_font_size *Adjust Theme Sizes in ggplot2*

Description

This function adjusts the sizes of various theme elements in a ggplot2 plot. It takes a single argument which is the base size for geom_text, and it calculates the theme sizes based on this.

Usage

```
tar_ggplot_font_size(geom_text_size = 7)
```

Arguments

geom_text_size	The base size for text elements in ‘geom_text‘. Default is 7.

Details

Always add the same size as in geom_text, then all fonts will have the same size. Default size is 7. If there is no geom_text we can control the font size of axis, legend, title and text with this function.

Value

A ‘theme‘ object that can be added to a ggplot.

References

The formula is from here: https://stackoverflow.com/questions/25061822/ggplot-geom-text-font-size-control

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(factor(vs), y = mpg, fill = factor(am))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "my_title") +
  tar_ggplot_font_size(10)
print(p)
```

---

tar_head_tail *Display both the head and tail of a dataframe or tibble*

Description

This function returns the first ‘nh‘ = n head and last ‘nt‘ = n tail rows of a dataframe or tibble. Row numbers from the original data are preserved as row names in the result.

Usage

```
tar_head_tail(data, nh = 5, nt = 5)
```

Arguments

data	A data frame or tibble whose head and tail you wish to view.
nh	An integer specifying the number of rows from the start of ‘data‘ to display. Default is 5.
nt	An integer specifying the number of rows from the end of ‘data‘ to display. Default is 5.

Value

A dataframe composed of the first ‘nh‘ and last ‘nt‘ rows of the input data. The row numbers from the original data are used as row names in the result.

See Also

rownames

Examples

```
if(interactive()){
 # Generate example data
 df <- data.frame(A = 1:10, B = 11:20)
 tar_head_tail(df, nh = 3, nt = 2)
 }
```

---

tar_impute_outlier *Imputates outliers and shows 4 different method plots*

Description

Imputates outliers and shows 4 different method plots

Usage

```
tar_impute_outlier(df, variable)
```

Arguments

df	data frame or tibble where the numeric variable lives
variable	Numeric variable column in the data frame

Value

4 plots with mean, median, mode and capping imputation

Examples

```
if (interactive()) {
  tar_impute_outlier(mtcars, mpg)
}
```

tar_install *Short Installer*

Description

Helper that returns an ‘install.packages()‘ command for a package name.

Usage

```
tar_install(x)
```

Arguments

x	The name of the package you want to install.

Details

This function uses non-standard evaluation to allow you to specify the package name unquoted.

Value

A character string with the corresponding ‘install.packages()‘ command.

Examples

```
tar_install(dplyr)
```

tar_median_survival *Plot Survival Curves and Save with Median Annotations*

Description

This function fits a survival model, plots the survival curves, and annotates median survival times. The plot can optionally be saved to file.

Usage

```
tar_median_survival(
  df,
  var,
  time_col = "time",
  status_col = "status",
  output_file = NULL
)
```

Arguments

df	A data frame containing the survival data.
var	A variable used for grouping the survival curves.
time_col	A string specifying the name of the column representing time.
status_col	A string specifying the name of the column representing status.
output_file	Optional output path for saving the plot. If 'NULL' (default), the function does not write a file.

Value

A 'ggsurvplot' object (invisibly). Optionally writes a PNG file.

Examples

```
# Example dataset
df_survival <- structure(
  list(
    status = c(1, 0, 1, 1, 1, 1, 0, 1, 1, 0),
    primorgan = structure(
      c(1L, 1L, 1L, 2L, 1L, 1L, 1L, 2L, 2L, 2L),
      levels = c("Colon", "Rectum"),
      class = "factor"
    ),
    sex = structure(
      c("male", "female", "male", "male", "male",
        "male", "male", "male", "male", "female"),
      label = "Gender"
    ),
    time = c(4.26, 49.52, 18.05, 11.04, 47.67, 8.03, 76.2, 15.44, 22.74, 50.64),
    subtype = structure(
      c(2L, 1L, 2L, 3L, 2L, 3L, 1L, 1L, 1L, 2L),
      levels = c("adenocarcinoma", "mucinous", "signet ring cell"),
      label = "Histological subtype",
      class = "factor"
    )
  ),
  row.names = c(NA, -10L),
  class = c("tbl_df", "tbl", "data.frame")
)

tar_median_survival(
  df = df_survival,
  var = sex,
  time_col = "time",
  status_col = "status",
  output_file = file.path(tempdir(), "survival_sex.png")
)
```

tar_oneline *tar_oneline*

Description

Creates a one-liner from a multiline code example in the clipboard.

Usage

```
tar_oneline()
```

Value

A character vector with the reformatted one-line code.

See Also

tidy_source

Examples

```
if (interactive()) {
  # Your multiline string
  # copy this code
  numericInput(
    "Register_Nr",
    "Nr.",
    min = 0,
    max = 20,
    step = 1,
    value = 0
  )
  # then apply
  tar_oneline()
}
```

tar_print *Set Pillar Print Minimum*

Description

This function sets the ‘pillar.print_min‘ option in R. This option controls globally the minimum number of data to print in tibbles. In contrast to data frames, in tibbles only 10 lines are shown except you use print(n=...). But you have to add it each time. tar_print can be use once globally and changed whenever necessayr!

Usage

```
tar_print(value = 50)
```

Arguments

value	A single non-negative integer for the minimum number of rows shown of a tibble in the console. Defaults to 50.

Value

Invisible null. The function is called for its side effect of setting an option.

References

The formula is from here: https://stackoverflow.com/questions/77708674/is-there-a-global-option-to-adjust-the-default-setting-of-tibbles-that-displays

Examples

```
tar_print(100)  # Set the pillar.print_min option to 100
tar_print()     # Set the pillar.print_min option back to default (50)
```

---

tar_psa *Peritoneal Surface Area Calculation*

Description

This function calculates 1 the given height and weight. It's used as a factor for further calculations.

Usage

```
tar_psa(height, weight)
```

Arguments

height	A numeric value representing the height in cm.
weight	A numeric value representing the weight in kg.

Value

A numeric value representing 1

Examples

```
tar_psa(180, 80) # Expected output: 199.6421 cm^2
```

tar_read_data *Read registry data*

Description

Return the in-memory ‘responses‘ data frame, if present.

Usage

```
tar_read_data()
```

Value

A data frame or NULL if not available.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_update_data *Update registry record*

Description

Update an existing record in ‘responses‘ by row name.

Usage

```
tar_update_data(data)
```

Arguments

data	A named list or data frame with registry fields.

Value

Invisible NULL.

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_update_inputs *Update Shiny inputs from record*

Description

Push a registry record into Shiny input controls.

Usage

```
tar_update_inputs(data, session)
```

Arguments

data	A one-row data frame with registry fields.
session	Shiny session.

Value

Invisible NULL.

See Also

updateTextInput, updateNumericInput, updateDateInput

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

tar_WellPanel *tar_WellPanel*

Description

Creates a shinydashboard box equivalent with shiny

Usage

```
tar_WellPanel(n)
```

Arguments

n	integer the gives the number of months in follow-up

Details

Builds a wellPanel containing follow-up inputs for the given month.

Value

returns a wellpanel a quasi shinydashboard box (without ShinyDashboard)

Examples

```
if(interactive()){
 #EXAMPLE1
 }
```

---

tar_wsl_pfad *Convert a Windows Path to a WSL Path*

Description

Converts a Windows path (for example, `C:/Users/...` or `C:\Users\...`) into a WSL path (`/mnt/c/Users/...` by default). Optionally copies the converted path to the clipboard.

Usage

```
tar_wsl_pfad(path = getwd(), copy_to_clipboard = TRUE, mount_root = "/mnt")
```

Arguments

`path`	A single Windows path string. Defaults to `getwd()`.
`copy_to_clipboard`	Logical. If `TRUE`, try to copy the resulting WSL path to the clipboard.
`mount_root`	Mount root used by WSL. Defaults to `"/mnt"`.

Value

A single character string with the converted WSL path.

Examples

```
tar_wsl_pfad("C:/Users/tarka/project", copy_to_clipboard = FALSE)
tar_wsl_pfad("C:\\Users\\tarka\\project", copy_to_clipboard = FALSE)
```

`tar_yes_no` *Convert "yes" and "no" to 1 and 0*

Description

This function takes a vector and converts all instances of "yes" to 1, "no" to 0, and leaves other values as NA. It's useful for converting categorical "yes"/"no" responses into a numeric format that can be used in statistical analysis.

Usage

```
tar_yes_no(x)
```

Arguments

`x`	A vector containing elements to be converted. Elements should be character strings potentially including "yes" and "no".

Value

A numeric vector where "yes" is replaced with 1, "no" with 0, and other values with NA.

Examples

```
test_vector <- c("yes", "no", "maybe", "yes", "no")
tar_yes_no(test_vector)
```

Index