Package: thekidsbiostats (via r-universe)

May 26, 2026

Title Report Templates and general use functions - The Kids Research Institute Australia, Biostatistics Team

Version 1.9.1

Description This package includes RMarkdown and ioslides templates that incorporate the The Kid's style guide and emulate the Institutes Web theme. Currently, templates are available to produce stand-alone HTML reports and ioslides presentations. These templates also include colour palates that will automatically reproduce The Kids style in R graphics. A number of helper functions and data-sets are also available. Currently, this is in transition from previous branding to new branding, with added functionality.

Date 2026-05-20

License MIT + file LICENSE

LazyData true

Depends R (>= 4.3)

Imports dplyr (>= 1.1.0), extrafont (>= 0.1), flextable (>= 0.9), fs (>= 1.6), ggfortify, ggplot2 (>= 3.4.0), glue (>= 1.8), gtsummary (>= 2.0), janitor (>= 2.1.0), labelled (>= 2.8.0), later, lifecycle, lubridate (>= 1.7.9), magrittr (>= 2.0.1), MASS, officer (>= 0.6.0), ordinal, patchwork (>= 1.3), purrr (>= 0.3.4), quantreg, rlang, rstudioapi (>= 0.17), shiny (>= 1.10), shinyjs (>= 2.1.0), shinyFiles (>= 0.9.0), shinyTree (>= 0.3), showtext (>= 0.9), stringr (>= 1.4.0), sysfonts (>= 0.8), systemfonts (>= 1.2), tidyverse (>= 2.0.0), tibble (>= 3.0.1), tidyr (>= 1.1.0), tidyselect (>= 1.1.0)

Suggests broom (>= 0.5.6), broom.helpers (>= 1.17.0), broom.mixed (>= 0.2.6), forcats (>= 0.5.0), ggeffects (>= 2.2.0), GGally (>= 2.1.2), ggstance (>= 0.3.4), gt (>= 1.0.0), here (>= 1.0.1), htmltools (>= 0.5.0.0), huxtable (>= 5.5.0), igraph (>= 1.2.5), jtools (>= 1.2.5), kableExtra (>= 1.1.0), knitr, mockery, openxlsx, ProjectTemplate (>= 0.10.3), readr (>= 1.3.1), repmis (>= 0.5), rmarkdown, rvest (>= 1.0.0), testthat (>= 3.0.0), tinytable (>= 0.8), vdiffr, withr (>= 2.4.2), XML (>=

3.99-0.3), spelling

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

URL https://github.com/the-kids-biostats/thekidsbiostats, https://the-kids-biostats.github.io/thekidsbiostats/

BugReports https://github.com/the-kids-biostats/thekidsbiostats/issues

Language en-GB

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libnode-dev libx11-dev zlib1g-dev

Repository https://the-kids-biostats.r-universe.dev

Date/Publication 2026-05-26 07:40:23 UTC

RemoteUrl https://github.com/The-Kids-Biostats/thekidsbiostats

RemoteRef HEAD

RemoteSha d0122be628dbd86943c81c9c7c5c02f18585270a

Contents

`checkbox_labels` *checkbox_labels*

Description

Apply factor labels to categorical responses (checkboxes) per REDCap data dictionary

Apply factor labels to categorical responses (checkboxes) per REDCap data dictionary

Usage

```
checkbox_labels(x, dict)
```


```
checkbox_labels(x, dict)
```


Arguments

`x`	checkbox `Variable / Field Name` per data dictionary
`dict`	REDCap data dictionary

Value

Named list of character objects

Named list of character objects

Examples

```
## Not run:

dat <- tibble(var___0 = c("0", "1"), var___1 = c("1", "0"))

dictionary <- tibble(`Variable / Field Name` = "var",
  `Choices, Calculations, OR Slider Labels` = "0, No | 1, Yes")
```

```
dat <- labelled::set_variable_labels(dat, .labels = checkbox_labels("var", dictionary))


## End(Not run)

## Not run:

dat <- tibble(var___0 = c("0", "1"), var___1 = c("1", "0"))

dictionary <- tibble(`Variable / Field Name` = "var",
  `Choices, Calculations, OR Slider Labels` = "0, No | 1, Yes")

dat <- labelled::set_variable_labels(dat, .labels = checkbox_labels("var", dictionary))


## End(Not run)
```

---

clean_REDCap *clean_REDCap*

Description

Clean a REDCap extract by applying factor levels and convert column classes per the REDCap data dictionary.

Usage

```
clean_REDCap(
  d,
  dict,
  numeric_date = FALSE,
  yesno_to_bool = FALSE,
  quiet = FALSE
)
```

Arguments

d	REDCap (data frame)
dict	REDCap data dictionary (data frame)
numeric_date	(default FALSE) set to TRUE if MS Excel has *helpfully* converted to a numeric date
yesno_to_bool	(default FALSE) convert factors with levels c("Yes", "No") to logical objects?
quiet	(default FALSE) pass quiet argument to lubridate functions

Value

cleaned data frame

Examples

```
## Not run:

dat <- tibble(var = c("0", "1"))

dictionary <- tibble(`Variable / Field Name` = "var",
  `Field Type` = "radio",
  `Field Label` = "Label",
  `Choices, Calculations, OR Slider Labels` = "0, No | 1, Yes",
  `Text Validation Type OR Show Slider Number` = NA)

(dat_clean <- clean_REDCap(dat, dictionary))


## End(Not run)
```

`create_project` *Create a new project structure*

Description

Creates a new project folder with subfolders, an RProj file, and optional Quarto report. Can optionally open the project in a new session and close the current one.

Usage

```
create_project(
  project_name,
  path = NULL,
  folders = c("data-raw", "data", "admin", "docs", "reports", "scripts"),
  ext_name = "html",
  create_report = FALSE,
  create_rproj = TRUE,
  open_project = TRUE,
  ...
)
```

Arguments

`project_name`	Name of the new project folder.
`path`	Parent directory for the project. If NULL, prompts the user.
`folders`	Character vector of subfolders to create.
`ext_name`	Name of the report extension for `create_template()`.
`create_report`	Whether to create a report using `create_template()`.
`create_rproj`	Whether to include a .Rproj file.
`open_project`	Whether to open the new project in a new RStudio session.
`...`	Additional arguments passed to `create_template()`.

Value

Invisibly returns the project path.

`create_project_addin` *Shiny Addin: Create New Project*

Description

Launches a Shiny app to create a project with folders, report, and control over session state.

Usage

```
create_project_addin()
```

`create_template` *Create a Quarto report template*

Description

Copies extension files into `_extensions/` and creates a `.qmd` report file. Optionally opens the created file. If the file already exists, no overwrite occurs and a warning is shown.

Usage

```
create_template(
  file_name = NULL,
  directory = "reports",
  ext_name = "html",
  title = NULL,
  subtitle = NULL,
  author = NULL,
  affiliation = NULL,
  include_reproducibility = TRUE,
  open_file = TRUE
)
```

Arguments

`file_name`	Name of the report file (without .qmd).
`directory`	Directory to save the report and extensions in.
`ext_name`	Name of the template extension (e.g. "html", "word").
`title`	Title of the report.
`subtitle`	Subtitle of the report.

author	Author name (defaults to Sys.info()[['user']]).
affiliation	Affiliation of the author.
include_reproducibility	Whether or not to include a 'Reproducibility Information' section, which outputs the R session information (default: TRUE).
open_file	Whether to open the file after creation (default: TRUE).

Value

Invisibly returns path to created .qmd file, or NULL if skipped.

---

create_template_addin *Shiny Addin: Create Report Template*

Description

Launches a Shiny app to generate a report template in a user-selected folder.

Usage

```
create_template_addin()
```

---

data_patient *Example of a (fictional) uncleaned data set for analysis*

Description

Includes variables that represent common patient information, with column names that would generally be problematic in an analytical setting without some form of cleaning.

Usage

```
data_patient
```

Format

data.frame

factor_convert *factor_convert*

Description

Apply factor labels to categorical responses per REDCap data dictionary.

Apply factor labels to categorical responses per REDCap data dictionary.

Usage

```
factor_convert(x, d, dict)
```


```
factor_convert(x, d, dict)
```


Arguments

x	character vector
d	REDCap import
dict	REDCap data dictionary

Examples

```
## Not run:

dat <- tibble(var = c("0", "1"))

dictionary <- tibble(`Variable / Field Name` = "var",
  `Choices, Calculations, OR Slider Labels` = "0, No | 1, Yes")

mutate(dat, across("var", factor_convert, d = dat, dict = dictionary))


## End(Not run)

## Not run:

dat <- tibble(var = c("0", "1"))

dictionary <- tibble(`Variable / Field Name` = "var",
  `Choices, Calculations, OR Slider Labels` = "0, No | 1, Yes")

mutate(dat, across("var", factor_convert, d = dat, dict = dictionary))


## End(Not run)
```

factor_table *factor_table*

Description

factor_table

factor_table

Usage

```
factor_table(x, rows = "\\|", cols = ",")

factor_table(x, rows = "\\|", cols = ",")
```

Arguments

x	character vector of REDCap options (e.g. 0, No | 1, Yes)
rows	(default "|") Regex defining row separator
cols	(default ",") Regex defining column separator

Details

For a more thorough example, see the vignette.

Value

2 column tibble of factor levels ("key") and labels ("value")

2 column tibble of factor levels ("key") and labels ("value")

Examples

```
## Not run:  factor_table("0, No | 1, Yes")

## Not run:  factor_table("0, No | 1, Yes")
```

`fct_case_when` *fct_case_when*

Description

Essentially the same as dplyr::case_when but the variable is of class factor, ordered based on the order entered into the case_when statement

Usage

```
fct_case_when(..., message = T)
```

Arguments

`...`	as per dplyr::case_when, essentially all input is passed through to dplyr::case_when A sequence of two-sided formulas. The left hand side (LHS) determines which values match this case. The right hand side (RHS) provides the replacement value. The LHS must evaluate to a logical vector. The RHS does not need to be logical, but all RHS must evaluate to the same type of vector. Both LHS and RHS may have the same length of either 1 or n. The value of n must be consistent across all cases. The case of n == 0 is treated as a variant of n != 1. NULL inputs are ignored.
`message`	prints the resulting factor levels. This is TRUE by default as output may often not be as expected

Details

For a more thorough example, see the vignette.

Value

A vector of length 1 or n, of class factor, matching the length of the logical input or output vectors. Inconsistent lengths or types will generate an error.

Examples

```
## Not run:

  x <- 1:50
  case_when(
    x %% 35 == 0 ~ "fizz buzz",
    x %% 5 == 0 ~ "fizz",
    x %% 7 == 0 ~ "buzz",
    TRUE ~ "no fizz buzz"
  ) %>% table
```

```
  fct_case_when(
    message = F,
    x %% 35 == 0 ~ "fizz buzz",
    x %% 5 == 0 ~ "fizz",
    x %% 7 == 0 ~ "buzz",
    TRUE ~ "no fizz buzz"
  ) %>% table


## End(Not run)
```

`insert_callout` *Insert Callout via RStudio Addin*

Description

Launches a Shiny app as an RStudio addin to insert a Quarto callout at the cursor position. The user selects a callout type from a dropdown, sees a preview colour, and inserts formatted callout text into the current script.

Usage

```
insert_callout()
```

Value

A Shiny application that runs within RStudio.

Examples

```
## Not run:
if (interactive()) {
  insert_callout()
}

## End(Not run)
```

insert_callout_2 *Insert Callout via RStudio Addin*

Description

Inserts a Quarto callout block with placeholder text directly at the cursor.

Usage

```
insert_callout_2()
```

Value

None. Inserts text into the active RStudio document.

Examples

```
## Not run:
insert_callout()

## End(Not run)
```

insert_margin *Insert Margin Block at Cursor*

Description

Inserts a Quarto column-margin block with placeholder text directly at the cursor.

Usage

```
insert_margin()
```

Value

None. Inserts text into the active RStudio document.

Examples

```
## Not run:
if (interactive()) {
  insert_margin()
}

## End(Not run)
```

insert_model_tabset *RStudio Addin: Insert child Quarto tabset for model output*

Description

This Shiny gadget inserts a `model_name <- "mod"` line and a child chunk to include a preformatted tabset from an external Quarto file. It also optionally previews the output.

Usage

```
insert_model_tabset()
```

Value

Inserts code into the active RStudio document and copies a child QMD file.

layout_params *Image Size Parameters for Save Function*

Description

Size parameters (mm) for a standard A4 page with "quarter", "half portrait", "half landscape", #' "full portrait", "full landscape". Identifiers for "pdf" and "png" devices included.

Usage

```
layout_params
```

Format

data.frame

`make_column_dict` *Create a copy-pasteable variable dictionary template*

Description

Create a copy-pasteable variable dictionary template

Usage

```
make_column_dict(
  data,
  quiet = FALSE,
  auto_clean = TRUE,
  new_names = NULL,
  labels = NULL,
  file = NULL,
  ...
)
```

Arguments

`data`	A data.frame or tibble containing the original data with column names to be replaced.
`quiet`	Logical, default FALSE: suppress printing the generated R code string to the console.
`auto_clean`	Logical, default TRUE: generate suggested new names using `janitor::clean_names()`
`new_names`	Optional character string, If specified, fills the `$new` column with these labels. Length must match the number of columns in `data`.
`labels`	Optional character string. If specified, fills the `$label` column with these labels. Length must match the number of columns in `data`.
`file`	Optional character string. If specified, writes dictionary template to file path (available formats: .csv, .txt, .xlsx, .rds, .R)
`...`	Additional arguments passed to `janitor::clean_names()`

Details

The default behaviour is to print a tribble template to the console. Set `quiet=TRUE` to suppress this output. If the file path has the extension .R, this writes the R code in the format of a `tribble` to the file.

Value

A data.frame (invisibly returned) with one row per column name in `data` and three columns:

old Original column names.

new Suggested new column names (optionally cleaned using `janitor::clean_names()`).

label Empty character field for user-supplied variable labels.

Examples

```
data("data_patient", package = "thekidsbiostats")

# Create a data dictionary and formulate some cleaned column names, assign to `dict` object
dict <- make_column_dict(data_patient, auto_clean = TRUE, quiet = FALSE)
```

`modify_labels` *Modify flextable Labels*

Description

Apply formatting and styling specifically to the label cells of a `flextable` object. Label cells are automatically identified as the cells with the smallest left padding in the specified column, which typically correspond to row labels.

Usage

```
modify_labels(
  x,
  label_col = 1,
  j = NULL,
  hline = NULL,
  bold = NULL,
  italic = NULL,
  color = NULL,
  bg = NULL,
  highlight = NULL,
  fontsize = NULL,
  font = NULL,
  rotation = NULL,
  align = NULL,
  border = NULL,
  border.top = NULL,
  border.left = NULL,
  border.bottom = NULL,
  border.right = NULL,
  padding = NULL,
  padding.top = NULL,
  padding.left = NULL,
  padding.bottom = NULL,
  padding.right = NULL
)
```

Arguments

x	A flextable object.
label_col	Numeric or character; the column used to identify label rows. Label rows are automatically detected as those with the smallest left padding in this column. Default is 1.
j	Numeric or character; the column(s) to which the styling/formatting should be applied, which by default is label_col.
hline	specifies a horizontal line above label cells. Can be an fp_border object, a color string (e.g., "red"), or a logical value (TRUE for default border, FALSE for no line). Note that j is ignored for hline, use border to apply to specific columns only.
bold	Logical; make label text bold.
italic	Logical; make label text italic.
color	Character; text colour of the label cells.
bg	Character; background colour of the label cells.
highlight	Character; highlight colour for the label cells.
fontsize	Numeric; font size of the label cells.
font	Character; font family name of the label cells.
rotation	Numeric; rotation of label text, can be one of "lrtb", "tbrl", "btlr".
align	Character; text alignment of label ("left", "center", "right" or "justify").
border, border.top, border.left, border.bottom, border.right	Optional fp_border objects to set cell borders.
padding, padding.top, padding.left, padding.bottom, padding.right	Optional numeric values to set cell padding.

Details

This function identifies label cells by finding the rows in column label_col with the smallest left padding. It then applies any formatting options specified and leaving unchanged those options that were left unspecified.

Horizontal lines specified via hline are applied above the label cells and across the entire table. If hline is a logical value, TRUE applies a default border and FALSE makes it transparent.

Value

A flextable object with the specified modifications applied to the label cells.

Examples

```
library(flextable)
df <- data.frame(Group = c("A","B","C"), Value = 1:3)
ft <- thekids_table(df)  # or flextable(df)
ft <- modify_labels(ft, bold = TRUE, hline = "black", fontsize = 12)
```

`preprocess_qmd` *Preprocess font changes to Quarto (.qmd) HTML reports*

Description

This function "pre-processes" a .qmd file to use different (Google) fonts, without having to locally download and install the font files.

Usage

```
preprocess_qmd(
  file_path,
  base_family = getOption("thekidsbiostats.font", "Barlow"),
  update_global_option = T
)
```

Arguments

`file_path`	File path to the .qmd file to update.
`base_family`	Font base family to consider. Defaults to the global option when loading `thekidsbiostats`. Otherwise, any Google font family can be specified (must be in `sysfonts::font_families_google()`).
`update_global_option`	Update the global option for `thekidsbiostats.font` based on the new font family specified. Default TRUE.

Value

Updates the YAML header of the input .qmd file (formatted per `thekidsbiostats` template) with the specified font `base_family`.

Note

Works only for HTML report types that have `mainfont` and `include-in-header` arguments in the YAML header as specified in the `thekidsbiostats` HTML template. The template can be created using thekidsbiostats::create_template.

Examples

```
## Not run:
preprocess_qmd(file_path = "path_to_qmd", base_family = "Monsieur La Doulaise")

## End(Not run)
```

round_df *round_df*

Description

A function, for data frames or tibbles, to round all numeric columns in a data frame or tibble.

Usage

```
round_df(x, digits = 2, con_char = F)
```

Arguments

x	a data frame or tibble vector
digits	integer indicating the number of decimal places to be used
con_char	logical as to whether you want the rounded columns to be converted to character class (T) or not (F); default is F

Details

The option exists to keep these columns as numeric (simply rounding), or to convert them to characters keeping trailing zeroes in the process.

Value

object of class tibble"

Examples

```
## Not run:
round_df(iris, 0)

round_df(iris, 0, con_char = T) %>% str

## End(Not run)
```

round_vec *round_vec*

Description

A function, for vectors, to keep trailing zeroes when rounding numbers.

Usage

```
round_vec(x, digits = 2)
```

Arguments

x	a numeric vector
digits	integer indicating the number of decimal places to be used.

Details

For a more thorough example, see the vignette.

Value

object of class character"

Examples

```
round_vec(mtcars$wt, 2)
```

scale_thekids *The Kids Colour Scales for ggplot2*

Description

Scale functions (fill and colour) for ggplot2.

Usage

```
scale_fill_thekids(
  palette = "primary",
  discrete = TRUE,
  reverse = FALSE,
  begin = 0,
  end = 1,
  na.value = thekids_colours$coolgrey_50,
```

```
  ...
)

scale_colour_thekids(
  palette = "primary",
  discrete = TRUE,
  reverse = FALSE,
  begin = 0,
  end = 1,
  na.value = thekids_colours$coolgrey_50,
  ...
)

scale_color_thekids(
  palette = "primary",
  discrete = TRUE,
  reverse = FALSE,
  begin = 0,
  end = 1,
  na.value = thekids_colours$coolgrey_50,
  ...
)
```

Arguments

palette	Name of the palette in thekids_palettes.
discrete	A boolean indicating whether fill aesthetic is discrete or not.
reverse	A boolean indicating whether the palette should be reversed.
begin	A numeric value between 0 and 1 at which the colour palette should begin.
end	A numeric value between 0 and 1 at which the colour palette should end.
na.value	A colour to use for missing values (Default='coolgrey_50').
...	Additional arguments passed to `discrete_scale()` and `scale_colour_gradientn()`/`scale_fill_gra` used respectively when discrete is TRUE or FALSE

---

thekids_colours *The Kids Research Institute Australia Colours*

Description

Colour names and HEX codes consistent with Institute guidelines.

Usage

```
thekids_colours
```

Format

A named character string. The following colours can be accessed with three variants (main colours and their 50% or 10% tint variations):

- Saffron: `saffron`, `saffron_50`, `saffron_10`
- Pumpkin: `pumpkin`, `pumpkin_50`, `pumpkin_10`
- Teal: `teal`, `teal_50`, `teal_10`
- Dark teal: `darkteal`, `darkteal_50`, `darkteal_10`
- Celestial blue: `celestialblue`, `celestialblue_50`, `celestialblue_10`
- Azure blue: `azureblue`, `azureblue_50`, `azureblue_10`
- Midnight blue: `midnightblue`, `midnightblue_50`, `midnightblue_10`
- Cool grey: `coolgrey`, `coolgrey_50`, `coolgrey_10`

Source

The Kids Research Institute Australia style guide.

Examples

```
names(thekids_colours)
thekids_colours$teal
thekids_colours$saffron_50
```

`thekids_model` *Fit a Statistical Model and Pass to Output Handler*

Description

This function fits a statistical model to the given dataset using a specified formula based on the dependent (`y`) and independent (`x` and `formula`) variables. After fitting, the model is passed to a dedicated output handler (`thekids_model_output`) for further processing or customization. Supported model types include linear regression, negative binomial regression, and quantile regression.

Usage

```
thekids_model(data, y, x, formula = "", model = "linear", ...)
```

Arguments

`data`	A data frame containing the variables to be used in the model.
`y`	Character string specifying the name of the dependent variable.
`x`	Character string specifying the name of the primary independent variable.
`formula`	Optional character string specifying additional terms to include in the formula. If not provided, the formula will be auto-generated as `y ~ x`.

model	Character string specifying the type of model to fit. Currently supported are "linear", "negbin", or "quantile".
...	Additional arguments passed to the underlying modeling functions.

Details

This function validates the specified model type and constructs a formula based on the supplied arguments. The data is reduced to complete cases for the variables included in the formula before fitting the model. The fitted model is passed to thekids_model_output for further handling, allowing for streamlined customization of model outputs.

For a more thorough example, see the vignette.

Value

Model output relating to the appropriate modeling function (e.g., lm, glm.nb, or rq). This is the same object returned by thekids_model_output if the latter is called directly.

See Also

thekids_model_output

Examples

```
# Example 1: Linear model
## Not run:
data(mtcars)
thekids_model(mtcars, y = "mpg", x = "wt", model = "linear")

## End(Not run)

# Example 2: Linear model with factor
## Not run:
thekids_model(mtcars %>% mutate(cyl = as.factor(cyl)), y = "mpg", x = "cyl", model = "linear")

## End(Not run)

# Example 3: Quantile regression
## Not run:
library(quantreg)
data(engel)
thekids_model(engel, y = "foodexp", x = "income", model = "quantile", tau = 0.5)

## End(Not run)
```

`thekids_model_output` *Generate Model Output for Specified Model Type*

Description

This function identifies the type of the fitted model and chooses the appropriate method to produce model output.

Usage

```
thekids_model_output(model, by = NULL, data = NULL, ...)
```

Arguments

`model`	A fitted model object produced by `thekids_model` or another modeling function.
`by`	The main predictor of interest.
`data`	A data frame containing the variables to be used in the model.
`...`	Additional arguments passed to the output method for the specific model type.

Details

This function acts as a wrapper, delegating the call to the appropriate output method based on the class of the model object. The actual behaviour is determined by the specific method implementation for the model type.

Value

Output specific to the fitted model type, as determined by the appropriate output method (e.g., summary or custom output).

Examples

```
# Assuming `model` is a fitted model object
# thekids_model_output(model)
```

`thekids_pal` *Generates a function, which when supplied to ggplot2 will return the appropriate colour palette, depending on the number of levels.*

Description

Generates a function, which when supplied to ggplot2 will return the appropriate colour palette, depending on the number of levels.

Usage

```
thekids_pal(palette, discrete = FALSE, reverse = FALSE)
```

Arguments

`palette`	Character name of palette in thekids_palettes
`discrete`	Boolean Indicating whether the palette is discrete or not (i.e. sequential or diverging)
`reverse`	Boolean indicating whether the palette should be reversed

Value

A palette function based off the thekids_palettes. When called with an integer `n`, it returns a character vector of `n` hex colour values. Designed for use with scale_colour_thekids(), scale_fill_thekids() and other ggplot2 scale functions.

`thekids_palettes` *The Kids Research Institute Australia Colour Palettes*

Description

Colour palettes using colours from The Kids Research Institute Australia.

Usage

```
thekids_palettes
```

Format

A nested list of colourmaps Palettes are stored in a nested list of palette variants, which can be accessed by: `thekids_palettes$<variant>$<name>`

- **Sequential palettes**: Continuous colour gradients for ordered data values
  - `thekids_palettes$sequential$primary`
  - `thekids_palettes$sequential$tint50`

  - `thekids_palettes$sequential$tint10`
  - mono-colour sequential ramps, accessed by `thekids_palettes$sequential$...` `saffron`, `pumpkin`, `teal`, `celestialblue`, `azureblue`, `midnightblue`, `coolgrey`

  These return palette functions of the form `function(n)` for continuous scales.

- **Diverging palettes**: Two-ended colour gradients deviating about a central white reference point.
  - `thekids_palettes$diverging$pumpkin2celestial`
  - `thekids_palettes$diverging$saffron2teal`
  - `thekids_palettes$diverging$saffron2midnight`

  These return palette functions of the form `function(n)` for diverging scales.

- **Qualitative palettes**: Sets of visually distinct colours for nominal (unordered) categories.
  - `thekids_palettes$qualitative$primary` - NOT YET IMPLEMENTED
  - `thekids_palettes$qualitative$tint50` - NOT YET IMPLEMENTED
  - `thekids_palettes$qualitative$tint10` - NOT YET IMPLEMENTED

  These return a named list of colours.

Details

In addition to the nested lists (sequential, diverging and qualitative), there are also three separate lists at the base level, namely: `$primary`, `$tint50`, `$tint10`. These historically contained the list of The Kids colours (which have now been moved to `thekids_colours`) but have been left here for backwards compatibility. Future updates of `thekidsbiostats` package will remove these lists from `thekids_palettes` object.

Source

The Kids Research Institute Australia style guide.

Examples

```
thekids_palettes$sequential$primary
thekids_palettes$sequential$midnightblue
thekids_palettes$diverging$saffron2teal
```

---

thekids_save *Save a ggplot with common practical defaults*

Description

This function saves ggplot2 plots to common pre-specified sizes and ratios.

Usage

```
thekids_save(
  plot = ggplot2::last_plot(),
  filename,
  path = ".",
  layout = "full landscape",
  device = "pdf",
  dpi = 300,
  ...
)
```

Arguments

plot	Plot to save; defaults to last plot displayed.
filename	File name to create on disk.
path	Path of the directory to save plot to. Defaults to current working directory.
layout	Size to save plot to. Can be a vector of strings. Default "full landscape". See details.
device	Device to use. Can be a vector of strings. Default "pdf".
dpi	Plot resolution. Default 300.
...	Other defaults passed to ggsave.

Details

Plot layouts can take the values "quarter", "half portrait", "half landscape, "full portrait", "full landscape". Each of these are based on the dimensions of a standard A4 page.

Examples

```
## Not run:
gg <- ggplot(mtcars, aes(hp, mpg, color = as.factor(cyl))) +
geom_point() +
theme_thekids()

thekids_save("example", layout = "half landscape", device = "png")
thekids_save("example", layout = "half portrait", device = "png")
thekids_save("example", layout = "full landscape", device = "png")

thekids_save("example", layout = c("half landscape", "half portrait"))
thekids_save("example", device = c("png", "pdf"))

## End(Not run)
```

thekids_showpalette *Function to display the available colours in The Kids palette*

Description

Displays the colour palette for The Kids, including their hex codes.

Usage

```
thekids_showpalette()
```

thekids_table *The Kids themed table output*

Description

A function that accepts tabular data, in a range of formats, and outputs an object of class `flextable()` (intended) for display in html and word documents.

Usage

```
thekids_table(
  x,
  fontsize = 10,
  fontsize_header = 11,
  line_spacing = 1.5,
  padding = 2.5,
  colour = "midnightblue",
  highlight = NULL,
  highlight_colour = NULL,
  zebra = FALSE,
  font_family = "Barlow",
  fallback_font_family = "sans",
  date_fix = TRUE,
  ...
)
```

Arguments

x	a table, typically a data.frame, tibble, or output from gtsummary.
fontsize	the font size for text in the body of the table, defaults to 8 (passed through to set_flextable_defaults).
fontsize_header	the font size for text in the header of the table, defaults to 10.

line_spacing	line spacing for the table, defaults to 1.5 (passed through to set_flextable_defaults).
padding	padding around all four sides of the text within the cell, defaults to 2.5 (passed through to set_flextable_defaults).
colour	a named colour, hex code or one of the colours from The Kids palette, see thekids_colours for available colour names.
highlight	A numeric vector indicating which rows to highlight. Defaults to NULL, meaning no rows are highlighted.
highlight_colour	Colour used when highlighting rows for both highlight and zebra. If NULL, a 50% tint of the main colour argument is used.
zebra	controls alternating highlighting of rows, logical or integer (defaults to FALSE); if TRUE, alternate each row's background with colour; if an integer, alternate row blocks of this size are highlighted; if negative, this will invert the sequence of highlighted blocks; (defaults to FALSE)
font_family	string containing the font family to apply to the table. Default "Barlow".
fallback_font_family	fallback font family if font_family is does not exist. Default is "sans".
date_fix	re-wraps date objects to strictly occupy one line, instead of splitting (defaults to TRUE).
...	other parameters passed through to set_flextable_defaults.

Details

The purpose of this function is easily coerce many different table structures in a consistent format (look and feel), with The Kids branding applied, that ultimately will look nice in either an html or word output.

Default settings produce a relatively compact table, to avoid reports becoming excessively lengthy.

The output can be piped (%>%) into further flextable() functions for advance customisation of the appearance.

Currently the function works well with input in the form of data frames, tibbles, dplyr pipes (think summarise()), gtsummary, and kable outputs.

For a more thorough example, see the vignette.

Value

a flextable class object that will display in both html and word output

Note

Errors may be encountered if the input to the function (kable/gtsummary/flextable) has already received a lot of processing (merging cells, aesthetic changes). The intention is that these things would occur after running thekids_table().

Font family must be installed at a system level, otherwise the default ("sans") will be applied.

Pre-specified formatting applied to 'flextable' objects (ahead of thekids_table()) may not carry over as expected. Please consider using thekids_table() in place of an explicit flextable() call, because our function already coerces the table to a flextable object.

Examples

```
## Not run:

head(mtcars, 10) %>%
  thekids_table(colour = "Saffron")


mtcars %>%
  select(cyl, mpg, hp, wt, gear) %>%
  group_by(cyl, gear) %>%
  summarise(mean_mpg = mean(mpg),
            mean_hp = mean(hp),
            mean_wt = mean(wt)) %>%
  thekids_table(colour = "CelestialBlue", highlight = c(4:6),
                padding.left = 20, padding.right = 20)

## End(Not run)
```

---

thekids_theme *Apply Institute Theming to ggplot2 Plots*

Description

This function applies a custom theme to ggplot2 plots, incorporating specific fonts and colours to align with the institute's visual identity.

Usage

```
thekids_theme(
  base_size = 11,
  base_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  strip_colour = "midnightblue",
  scale_colour_type = lifecycle::deprecated(),
  scale_fill_type = lifecycle::deprecated(),
  colour_theme = lifecycle::deprecated(),
  fill_theme = lifecycle::deprecated(),
  rev_colour = lifecycle::deprecated(),
  rev_fill = lifecycle::deprecated(),
  fig_dpi = 300,
  ...
)

theme_thekids(
  base_size = 11,
  base_family = NULL,
```

```
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  strip_colour = "midnightblue",
  scale_colour_type = lifecycle::deprecated(),
  scale_fill_type = lifecycle::deprecated(),
  colour_theme = lifecycle::deprecated(),
  fill_theme = lifecycle::deprecated(),
  rev_colour = lifecycle::deprecated(),
  rev_fill = lifecycle::deprecated(),
  fig_dpi = 300,
  ...
)
```

Arguments

`base_size`	The base font size, given in points. Default is 11.
`base_family`	The base font family used for the text (default Barlow). Most Google Fonts are supported (see Note).
`base_line_size`	The base size for line elements (e.g., axis lines, grid lines). Calculated as `base_size/22` by default.
`base_rect_size`	The base size for rect elements (e.g., plot background, legend keys). Calculated as `base_size/22` by default.
`strip_colour`	A named colour from the list of The Kids colours, see thekids_colours for available colour names. Note that only the main colour names are accepted because the 50% tint is always used.
`scale_colour_type`	Deprecated. Use `viridis_pal` instead.
`scale_fill_type`	Deprecated. Use `viridis_pal` instead.
`colour_theme`	Deprecated. Use `viridis_pal` instead.
`fill_theme`	Deprecated. Use `viridis_pal` instead.
`rev_colour`	Deprecated. Use `viridis_pal` instead.
`rev_fill`	Deprecated. Use `viridis_pal` instead.
`fig_dpi`	Base DPI for figure. Only applicable when Barlow font family (default) is *not* selected.
`...`	Miscellaneous arguments necessary for parameter aliasing, etc.

Details

The function determines the operating system and selects appropriate font names for Windows or other systems. It applies a minimal theme with custom settings for plot title, axis title, and strip text, using the 'Barlow Semi Condensed' font family. It also adjusts color scales using the 'viridis' package.

For a more thorough example, see the vignette.

Value

A list of ggplot2 theme elements and scale adjustments.

Note

If a Google font has not been loaded with the package, `thekids_theme` will load this function on your behalf.

Supported Google Fonts are those that exist in `sysfonts::font_families_google()`.

Examples

```
## Not run:
# Install the required fonts first (see below)
# Example usage with ggplot2
library(ggplot2)

p <- ggplot(mtcars, aes(x = mpg, y = wt, col = factor(cyl))) +
  geom_point() +
  thekids_theme() +
  scale_colour_thekids()

print(p)

p2 <- ggplot(mtcars, aes(x = factor(cyl), y = wt, fill = factor(cyl))) +
  geom_col() +
  thekids_theme() +
  scale_fill_thekids(palette='tint50', reverse=TRUE)

print(p2)

## End(Not run)
## Not run:
# Install the required fonts first (see below)
# Example usage with ggplot2
library(ggplot2)

p <- ggplot(mtcars, aes(x = mpg, y = wt, col = factor(cyl))) +
  geom_point() +
  theme_thekids() +
  scale_colour_thekids()

print(p)

p2 <- ggplot(mtcars, aes(x = factor(cyl), y = wt, fill = factor(cyl))) +
  geom_col() +
  theme_thekids() +
  scale_fill_thekids(palette='tint50', reverse=TRUE)

print(p2)

## End(Not run)
```

update_columns *Apply a variable dictionary to a dataset. Each column is replaced with a new name and a corresponding label attribute is applied.*

Description

Apply a variable dictionary to a dataset. Each column is replaced with a new name and a corresponding label attribute is applied.

Usage

```
update_columns(
  data,
  dict,
  old = NULL,
  new = NULL,
  label = NULL,
  reorder = FALSE
)
```

Arguments

data	A data.frame-like object
dict	A data.frame-like object or .csv file path (.csv, .txt, .xlsx, .rds). Should have 3 columns specifying: old names, new names, and labels.
old	Column name in dict with old column names (default "old")
new	Column name in dict with new column names (default "new")
label	Column name in dict with human-readable labels (default "label")
reorder	Logical, default FALSE. If TRUE, columns listed in the dictionary will be reordered according to the order they appear in the dictionary. Columns not referenced in the dictionary remain at the end in their original order.

Value

A data.frame with renamed columns and labels applied

Examples

```
data("data_patient", package = "thekidsbiostats")

# 1) Create a data dictionary and formulate some cleaned column names, assign to `dict` object
dict <- make_column_dict(data_patient, auto_clean = TRUE, quiet = FALSE)

# 2) Apply column names to data, assign to `data_patient_clean`
data_patient_clean <- update_columns(data_patient, dict = dict)
```

variable_labels *variable_labels*

Description

Apply variable labels to a data frame per the REDCap data dictionary. If there is no variable label associated with a column then is lable will be the same as the column's name.

Apply variable labels to a data frame per the REDCap data dictionary. If there is no variable label associated with a column then is label will be the same as the column's name.

Usage

```
variable_labels(d, dict)

variable_labels(d, dict)
```

Arguments

d	REDCap import
dict	REDCap data dictionary

Value

character vector (length `ncol(d)`) of REDCap Field Labels per the data dictionary

data frame with variable labels

Examples

```
## Not run:

dat <- tibble(var = 1)

dictionary <- tibble(`Variable / Field Name` = "var", `Field Label` = "Label")

dat <- labelled::set_variable_labels(dat, .labels = variable_labels(dat, dictionary))


## End(Not run)

## Not run:

dat <- tibble(var = 1, var2___1 = 1, var2___2 = 0)
dictionary <- tibble(`Variable / Field Name` = c("var", "var2"),
                     `Field Label` = c("Label for field 'Var'", NA),
                     `Field Type` = c(NA, "checkbox"),
                     `Choices, Calculations, OR Slider Labels` =
                       c(NA, "1, First checkbox label | 2, Second checkbox label"))
```

```
dat <- variable_labels(dat, dictionary)
View(dat)


## End(Not run)
```

---

yesno_vars *yesno_vars*

Description

yesno_vars

Usage

```
yesno_vars(d)
```

Arguments

d	a data frame object

Value

a character vector of columns that are factors with levels: `c("Yes", "No")`

Examples

```
  ## Not run:

dat <- tibble(var = factor(c("Yes", "Yes", "No", "Yes"), levels = c("Yes", "No")))

yesno_vars(dat)

mutate(dat, across(yesno_vars(dat), ~case_when(. == "Yes" ~ TRUE, . == "No" ~ FALSE, TRUE ~ NA)))


## End(Not run)
```

Index