Package: WAACHShelp (via r-universe)

November 1, 2026

Version 1.5.3

Date 2026-02-27

Title Various helper functions to aid reproducibility in WAACHS analysis-related tasks

Author Zac Dempsey [aut, cre]

Maintainer Zac Dempsey <zac.dempsey@thekids.org.au>

Depends R (>= 4.4), tidyverse (>= 2.0.0), gtsummary (>= 2.4.0)

Imports colorspace (>= 2.1.0), data.table (>= 1.17.0), flextable (>= 0.9.0), ggplot2 (>= 3.5.0), grDevices, magrittr (>= 2.0.0), rlang (>= 1.1.0), table1, tibble (>= 3.2.0)

Suggests dplyr (>= 1.1.0), gt (>= 1.0.0), forcats (>= 1.0.0), haven, knitr, kableExtra, lifecycle (>= 1.0.0), lubridate (>= 1.9.0), mockery, officer, purrr (>= 1.1.0), readr (>= 2.1.0), rmarkdown, rstudioapi, rvest, sjmisc, spelling, stringr (>= 1.5.0), systemfonts, tcltk, tidyr (>= 1.3.0), tidyselect (>= 1.2.0), utils, testthat (>= 3.0.0), withr (>= 3.0.0)

Description Data analysis requirements for the WAACHS project often involves a range of complex and nuanced functions. This is conflated by different analysts working across similar tasks at different points in time. This (local) package aims to ameliorate this with a set of functions applicable to all analysts (who use R)---aiding reproducibility.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

LazyData true

Config/testthat/edition 3

URL https://github.com/The-Kids-Biostats/WAACHShelp, https://the-kids-biostats.github.io/WAACHShelp/

BugReports https://github.com/The-Kids-Biostats/WAACHShelp/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 2026-02-27 06:29:07 UTC

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

RemoteRef HEAD

RemoteSha 0c5d89ae6ff8cc1b093aad6fca56cd27d08ad632

Contents

add_rows2 . . . 2
aducust_flag . . . 3
check_att . . . 5
create_markdown . . . 6
create_project . . . 7
icd_dat . . . 8
icd_morb_flag . . . 9
person_summary . . . 12
proc_contents . . . 13
proc_freq . . . 14
save_waachs . . . 15
sum_tab . . . 15
sumfun . . . 16
theme_waachs . . . 16
transpose_tblsum . . . 17
twoway . . . 18
val_filt . . . 19
waachs_palette . . . 19
waachs_table . . . 20

Index **22**

`add_rows2` *Add rows to labelled data set.*

Description

[Deprecated]

This function was deprecated because it was no longer required by analysts.

Usage

```
add_rows2(..., id = NULL)
```

Arguments

`...`	Other parameters to parse to function.
`id`	Default NULL

Details

Almost identical to the original `dplyr::add_row` but also looks for `format.sas` attribute which `haven` provides when loading as SAS dataset.

Created by PV (2023).

Value

SAS labelled dataframe

---

aducust_flag *Custodial data flagging function*

Description

This function flags whether a carer record (custodial, per its name) exists when a child is of a certain age.

Usage

```
aducust_flag(
  data,
  dobmap,
  carer_map,
  flag_name = "carer_aducust",
  child_id_var = "rootnum",
  carer_id_var = "carer_rootnum",
  data_start_date = "ReceptionDate",
  data_end_date = "DischargeDate",
  dobmap_dob_var = "dob",
  child_start_age = 0,
  child_end_age = 18,
  carer_summary = FALSE,
  any_carer_summary = FALSE
)
```

Arguments

`data`	Input dataset (carer aducust).
`dobmap`	DOBmap file at the child level.
`carer_map`	Mapping file with columns "child ID", "carer ID". Can have multiple rows per child (e.g., one per carer 1, carer 2, NEWBMID).
`flag_name`	Name of flagging variable to return. Default `"carer_aducust"`.
`child_id_var`	Variable denoting "child ID". Must exist and be called the same thing in `dobmap` and `carer_map`. Default `"rootnum"`.
`carer_id_var`	Variable denoting "carer ID". Must exist in `carer_map`. Default `"carer_rootnum"`.
`data_start_date`	Start date to consider in `data`. Corresponds to aducust *start* date. Default `"ReceptionDate"`.
`data_end_date`	End date to consider in `data`. Corresponds to aducust *end* date. Default `"DischargeDate"`.
`dobmap_dob_var`	Date of birth (DOB) variable in `dobmap`. Default `"dob"`.
`child_start_age`	Numeric. Start (minimum) age (years) to consider for flagging (default `0`).
`child_end_age`	Numeric. End (maximum) age (years) to consider for flagging (default `18`).
`carer_summary`	Collapse aducust flags *within carer* (i.e., for each `carer_id_var`). Default `FALSE`.
`any_carer_summary`	Collapse aducust flags *across carers*. Default `FALSE`.

Details

While it is designed for use with flagging carer custodial records, it can be applied in many other circumstances where flagging of a carer (or otherwise) record exists when a child (or otherwise) is of a certain age.

For more details, see the vignette.

Value

Flagged dataframe.

Note

- If `carer_summary` is `TRUE`, then we are flagging whether a specific carer has any aducust record.
- Therefore, we are assessing whether a specific carer has any aducust records.
- If `any_carer_summary` is `TRUE`, then we are flagging whether any carer (if multiple) have any aducust records.
- If `any_carer_summary` is `TRUE`, `carer_summary` will be ignored.

Examples

```
## Not run:
# Example 1: Basic use
aducust_flag(data = carer_aducust %>% rename(carer_rootnum = root),
             dobmap = dobmap,
             carer_map = child_carer_map,
             child_id_var = "NEWUID",
             carer_id_var = "carer_rootnum",
             carer_summary = FALSE,
             any_carer_summary = TRUE
)

## End(Not run)
```

---

check_att *Education flagging function*

Description

This function serves to flag whether an individual is expected to have education data for every year within a "visibility window" based on their date of birth. The fact that "staggered" (June/July) years were introduced in Western Australia in 1997 has been built into the function.

Usage

```
check_att(
  dob,
  visibility_min = 2008,
  visibility_max = 2014,
  show_expectation = TRUE
)
```

Arguments

dob	Date of individual. Must be class "Date".
visibility_min	Minimum year in visibility window. Default 2008.
visibility_max	Maximum year in visibility window. Default 2014.
show_expectation	Logical. Show flagged dataframe with expected flag per year in visibility window. Default TRUE.

Value

Flagged dataframe.

Examples

```
check_att(dob = as.Date("2000-05-30"))
```

`create_markdown` *Create Template for Quarto Markdown*

Description

This function creates a HTML or Word template in the current working directory using a specified Quarto extension. It copies the template files to the `_extensions/` directory and generates a new Quarto markdown (.qmd) file.

Usage

```
create_markdown(file_name = NULL, directory = "reports", ext_name = "html")
```

Arguments

`file_name`	A string. The name of the new Quarto markdown (.qmd) file. This must be provided.
`directory`	A string. The name of the directory to plate the files. Default is NULL. Requires user specification
`ext_name`	A string. The extension type to create. Default "html" (alternatives: "word").

Details

Adapted from create_template from `https://github.com/The-Kids-Biostats/thekidsbiostats`.

The function first checks whether a `_extensions/` directory exists in the current working directory. If not, it creates one. It then copies the necessary extension files from the package's internal data to the `_extensions/` directory. Finally, it creates a new Quarto markdown file based on the extension template.

By default, the `reports` folder will be selected to house the report.

For more details, see the vignette.

Note

The function assumes that the package `WAACHShelp` contains the necessary extension files under `ext_qmd/_extensions/`.

Examples

```
## Not run:
create_markdown(file_name = "my_doc", ext_name = "word")

## End(Not run)
```

create_project *Create a new project structure with extensions*

Description

Adapted from create_project from https://github.com/The-Kids-Biostats/thekidsbiostats.

Usage

```
create_project(
  project_name = "standard",
  data = TRUE,
  reports = TRUE,
  output = TRUE,
  documentation = TRUE,
  other_folders = NULL,
  R = TRUE
)
```

Arguments

project_name	String. Default "standard". Name of R Project object to be created.
data	Logical. If TRUE, a data directory will be created in the selected folder. Defaults to TRUE.
reports	Logical. If TRUE, a reports directory will be created in the selected folder. Defaults to TRUE.
output	Logical. If TRUE, an output direction will be created in the selected folder. Defaults to TRUE.
documentation	Logical. If TRUE, a documentation directory will be created in the selected folder. Defaults to TRUE.
other_folders	Vector of strings that contain any other folders that should also be created. Elements should be unique. Default NULL.
R	Logical. If TRUE, an R directory will be created in the selected folder. Defaults to TRUE.

Details

This function creates a directory structure for a new project based on a specified extension. It can also create additional folders such as data, reports, output and documentation. The function copies specific files and folders from the chosen extension to the project directory.

For more details, see the vignette.

Note

An interactive window will appear prompting the user to select the folder where the project structure should be created. The default is the current working directory.

Examples

```
## Not run:
create_project(project_name = "investigation_x", other_folders = c("folder1", "folder2"))

## End(Not run)
```

`icd_dat` *ICD Classification Data set*

Description

Data set specifying the ICD (9 & 10) codes for different events in the morbidity data set.

Usage

```
data(icd_dat)
```

Format

A data frame where rows correspond to an event and columns correspond to the variable name, morbidity search parameters (diagnosis/ediag, ecode) and ICD code breakdown

num Counter variable representing the number of rows associated with any given var

classification Classification type

var Variable name

broad_type Type of diagnosis field this ICD code corresponds to (diagnosis & ediag = "diag_ediag", ecode = "ecode")

letter Letter of ICD code (purely numeric is empty string "")

lower Lower bound of numeric element of ICD code

upper Upper bound of numeric element of ICD code ...

Source

Generated internally by package

`icd_morb_flag` *ICD flagging function*

Description

This function serves to add flags to an input data set (morbidity at this stage) pursuant to ICD codes. Flags can be added for general categories based on pre-established ICD codes (e.g., any mental health contact, any substance-related contact, any alcohol/tobacco-related contact etc.) or add a custom set of ICD codes. The file with these pre-established ICD codes are saved as an .RData file and are trivial to change.

Usage

```
icd_morb_flag(
  data,
  dobmap = NULL,
  flag_category,
  flag_other_varname,
  diag_type,
  diag_type_custom_vars = NULL,
  diag_type_custom_params,
  under_age = FALSE,
  age = 18,
  person_summary = FALSE,
  id_var = "rootnum",
  morb_date_var = "subadm",
  dobmap_dob_var = "dob",
  dobmap_other_vars = NULL
)
```

Arguments

`data`	Input dataset (morbidity).
`dobmap`	DOBmap file corresponding to input dataset.
`flag_category`	Type of flag to generate. Takes values from reference file (e.g., MH_morb, Sub_morb, etc.) or "Other" for custom ICD specification and flagging.
`flag_other_varname`	Flag variable name (specified only When `flag_category == "Other"`). Input as character string.
`diag_type`	Diagnosis type. Select from "principal diagnosis", (all) "additional diagnoses", "external cause of injury", "custom".
`diag_type_custom_vars`	Variables to search across when `diag_type == "custom"`.
`diag_type_custom_params`	Search parameters to search across when `diag_type == "custom"`. Must be a list where the keys are the variable names and values are the inputs to `WAACHShelp::val_filt`.

	Can also be a list of lists where multiple ICD can be searched across for a single variable. See examples for specification.
under_age	Return additional variables corresponding to when participant was strictly under age y.o.. Uses DOBmap DOB and subadm morbidity admission date. Variables have suffix "_under{age}".
age	Numeric. Age (years) to consider for the under_age variable (default 18).
person_summary	Summarise results at a person-level.
id_var	Joining (ID) variable consistent between data and dobmap. Default rootnum.
morb_date_var	Hospital morbidity date variable in data. Default "subadm".
dobmap_dob_var	Date of birth (DOB) variable in dobmap. Default "dob".
dobmap_other_vars	Other variables to carry across from DOBmap file when joining to data. Default NULL. Can be a vector of strings.

Details

For more details, see the vignette.

Value

Flagged dataframe.

Examples

```
## Not run:
# Example 1: Basic use
## Create any mental health or substance-related morbidity flag, "MH_morb"
## Searches "principal diagnosis", "additional diagnoses", "external cause of injury".
## Create additional flag for whether admission occurred when under 18 years of age
icd_morb_flag(
  data = morb,
  dobmap = dob,
  flag_category = "MH_morb",
  under_age = T,
  age = 18,
  dobmap_other_vars = c("xyz123", "abc456") # Also join `xyz123`, `abc456` from DOBmap
  )

# Example 2: Basic use
## Create any substance-related morbidity flag, "Sub_morb"
icd_morb_flag(data = morb,
              flag_category = "Sub_morb" # Create any MH contact flag
              )

# Example 3: Search  *principal diagnosis* and *first additional diagnosis*
# for a custom set of ICD codes
## Call this variable "test_var"
icd_morb_flag(data = morb,
              flag_category = "Other",
```

```
                diag_type = "custom",
                diag_type_custom_vars = c("diagnosis", "ediag20"),
                diag_type_custom_params = list("diagnosis" = list("letter" = "F",
                                                                  "lower" = 0,
                                                                  "upper" = 99.9999),
                                               "ediag20" = list("letter" = "",
                                                                "lower" = 0,
                                                                "upper" = 99.9999)),
                flag_other_varname = "test_var"
                )

    # Example 4:
    # Search only across primary diagnosis and
    # (all) additional diagnosis fields for a custom set of ICD codes.
    ## Call this variable "test_var2"
    icd_morb_flag(data = morb,
                flag_category = "Other",
                diag_type = c("principal diagnosis", "additional diagnoses"),
                diag_type_custom_params = list("principal diagnosis" = list("letter" = "F",
                                                                            "lower" = 0,
                                                                            "upper" = 99.9999),
                                               "additional diagnoses" = list("letter" = "",
                                                                             "lower" = 0,
                                                                             "upper" = 99.9999)),
                flag_other_varname = "test_var2"
                )

    # Example 5: Search across (all) additional diagnosis fields and another random field, `dagger`
    ## Call this variable "test_var3"
    icd_morb_flag(data = morb,
                flag_category = "Other",
                diag_type = c("custom", "additional diagnoses"),
                diag_type_custom_vars = "dagger",
                diag_type_custom_params = list("dagger" = list("letter" = "F",
                                                               "lower" = 0,
                                                               "upper" = 99.9999)),
                flag_other_varname = "test_var3"
                )

    # Example 6: Searching across multiple ICD code types within a variable
    ## Call this variable "test_var4" -- replicating MH_morb flag
    icd_morb_flag(data = morb,
                flag_category = "Other",
                diag_type = c("additional diagnoses",
                              "additional diagnoses",
                              "external cause of injury"),
                flag_other_varname = "test_var3",
                diag_type_custom_params =
                list("principal diagnosis" = list(list("letter" = "F",
                                                       "lower" = 0,
                                                       "upper" = 99.9999),
                                                  list("letter" = "",
                                                       "lower" = 290,
```

```
                                                "upper" = 319.9999)),
                        "additional diagnoses" = list(list("letter" = "F",
                                                           "lower" = 0,
                                                           "upper" = 99.9999),
                                                      list("letter" = "",
                                                           "lower" = 290,
                                                           "upper" = 319.9999)),
                        "external cause of injury" = list(list("letter" = "E",
                                                               "lower" = 950,
                                                               "upper" = 959.9999),
                                                          list("letter" = "X",
                                                               "lower" = 60,
                                                               "upper" = 84.9999))))

## End(Not run)
```

`person_summary` *Summarise a dataframe to a person level*

Description

In cases where we have a *long* data frame (i.e., multiple rows of data per participant) with a flag against each record (e.g., variable `x` = "Yes"/"No"), this function will collapse this dataframe to a participant level.

Usage

```
person_summary(data, flag_category, flag_category_val = "Yes", grouping_var)
```

Arguments

`data`	Input dataframe.
`flag_category`	Name of the variable to perform classification on.
`flag_category_val`	String value of the `flag_category` variable the flag will be judged against. Defaults to "Yes".
`grouping_var`	Grouping ID variable that identifies potentially multiple records per participant.

Details

The collapsing is based on the value of `flag_category_val` and a grouping variable (participant ID) `grouping_var`.

Specifically, records will be collapsed such that:

- If *any* record(s) for a participant is equal to `flag_category_val`, then return "Yes".
- If *all* record(s) for a participant are not equal to `flag_category_val`, then return "No".

Examples

```
## Not run:
person_summary(data = dat,
               flag_category = "variable_x",
               flag_category_val = "Yes",
               grouping_var = "record_id"
               )

## End(Not run)
```

`proc_contents` *Compile metadata from dataframe*

Description

[Deprecated]

This function was deprecated because it was no longer required by analysts.

Usage

```
proc_contents(df)
```

Arguments

`df`	Dataframe to input.

Details

This little function pulls out the labels and formats from a dataframe and compiles this metadata as a dataframe.

Imitates the "proc contents" function of SAS.

Created by PV (2023).

Value

Dataframe

Examples

```
proc_contents(iris)
```

proc_freq *HTML summary table*

Description

[Deprecated]

This function was deprecated because it was no longer required by analysts.

Usage

```
proc_freq(var1, data = NULL, sort = NULL, min.frq = 0)
```

Arguments

var1	Name of the variable
data	Dataset that contains var1
sort	Optional argument which can take on "asc" or "desc" to indicate the type of sort required.
min.frq	Minimum frequency

Details

Renders a HTML table ready for plonking into a HTML or word document.

Renders output similar to the "proc freq" function of SAS.

Created by PV (2023).

Value

Dataframe

Examples

```
test = iris
proc_freq(Species, test)
test[1:4, "Species"] <- NA
proc_freq(Species, test)
```

save_waachs *Save dataframe*

Description

General function to save a dataset in usable formats across different platforms. Specifically, will return .csv, .sas7bdat, and .RDS files.

Usage

```
save_waachs(dataframe, path, filename)
```

Arguments

dataframe	Input dataset to save.
path	Path to save file to.
filename	Name of the file to save.

Value

Saved object

sum_tab *Summary tabling function*

Description

General tabling function to standardise output and formatting across analysts and projects. Based on gtsummary::tbl_summary

Usage

```
sum_tab(data, ...)
```

Arguments

data	Input dataset.
...	Any other argument relevant to `gtsummary::tbl_summary`.

Value

Summary table.

sumfun *Summary function reminiscent of Stata's "tabset".*

Description

[Deprecated]

This function was deprecated because it was no longer required by analysts.

Usage

```
sumfun(x, na.rm = TRUE, ...)
```

Arguments

x	Vector from input dataframe to summarise.
na.rm	(Default TRUE) to remove NA (missing) observations from summary.
...	Any other arguments parsed into component base R functions.

Details

Created by PV (2023).

Value

Summary table with n (length), miss (number of missing observations), mean, sd (standard deviation), med (median), q25 (first quartile), q75 (third quartile), min (minimum value), max (maximum value).

Examples

```
sumfun(iris$Sepal.Length)
```

theme_waachs *Apply WAACHS theme to ggplot2 plots*

Description

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

Usage

```
theme_waachs(
  base_size = 12,
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
```

Arguments

base_size	Base font size
base_line_size	Base line size (default base_size/22)
base_rect_size	Base rectangle size (default base_size/22)

Details

The function determines the operating system and selects appropriate font names for Windows or other systems. It also adjusts colour scales.

Value

A list of ggplot2 theme elements and scale adjustments.

Examples

```
ggplot(mtcars,
       aes(x = mpg, y = wt, col = factor(cyl))) +
  geom_point() +
  theme_waachs()
```

---

transpose_tblsum *WAACHS transpose* gtsummary::tbl_summary *table*

Description

Quick helper function to easily transpose gtsummary::tbl_summary tables. Best used in conjunction with WAACHShelp::waachs_table.

Usage

```
transpose_tblsum(tbl, ...)
```

Arguments

tbl	tbl_summary input (of class c("tbl_summary", "gtsummary")).
...	Other parameters (not currently in use).

Value

data.frame with transposed summary table.

Examples

```
mtcars %>%
  mutate(cyl = as_factor(cyl)) %>%
  select(cyl, mpg, disp, hp, wt, am) %>%
  tbl_summary(by = cyl) %>%
 modify_header(label ~ "cyl") %>% # Re-label "Characteristic" by stratification variable ("cy")
  transpose_tblsum() %>%
  waachs_table()
```

`twoway` *HTML summary table*

Description

[Deprecated]

This function was deprecated because it was no longer required by analysts.

Usage

```
twoway(var1, var2, data = NULL, var2lab = NULL)
```

Arguments

`var1`	Vector of first variable
`var2`	Vector of second variable
`data`	Dataset containing `var1` and `var2`
`var2lab`	Label for `var2`.

Details

Two-way table similar to the "proc freq" function of SAS with two variables

Created by PV (2023).

Value

Two-way table

val_filt *Basic ICD code function*

Description

This function is already used by the team, and filters alphanumeric ICD-9 and ICD-10 codes pursuant to requirements.

Usage

```
val_filt(input_vec, letter, lower, upper)
```

Arguments

input_vec	Vector of all admissible ICD codes to filter on.
letter	Letter to base filtration on (if purely numeric use empty string "")
lower	Lower bound on the numeric element of the ICD code (includes numerics >=lower).
upper	Upper bound on the numeric element of the ICD code (includes numerics <=upper).

Value

Vector with filtered ICD codes.

Examples

```
# Filter ICD codes in val3 to those between F10 and F10.9 (inclusive).
## Not run:
val_filt(val3, "F", 10.0, 10.9)

## End(Not run)
```

waachs_palette *WAACHS colour palette*

Description

Function to return discrete or continuous colour palette based on WAACHS logo colours.

Usage

```
waachs_palette(
  type = "discrete",
  n,
  visualisation = F,
  bias = 2,
  interpolate = "spline",
  ...
)
```

Arguments

type	Type of colour palette to render (values "discrete", "continuous").
n	Number of colours to generate in palette (if type == "continuous").
visualisation	(Default TRUE). Return visualisation of the colour spectrum to aid in decision making.
bias	A positive number representing the spacing between colours at the high end. Parsed to grDevices::colorRamp (default 2).
interpolate	Interpolation algorithm to parse to grDevices::colorRamp (default "spline").
...	Miscellaneous arguments to parse to grDevices::colorRampPalette.

Value

Vector with colour hex codes. If type == "continuous" returns vector of length n. If visualisation == TRUE returns a list containing colour palette vector and ggplot2 visualisation of colour spectrum.

Examples

```
colours <- waachs_palette(type = "continuous",
                          n = 100,
                          visualisation = TRUE) # Render colour palette
print(colours$palette) # Print colour palette
print(colours$plot)    # Print plot
```

---

waachs_table *WAACHS table formatting function.*

Description

Function to apply consistent formatting to summary tables rendered using R. Works with functions from the gtsummary package, or dataframes.

Usage

```
waachs_table(
  x,
  font.size = 10,
  font.size.header = 11,
  line.spacing = 1.5,
  padding = 2.5,
  body_bg_col = "#FEF0D8",
  header_bg_col = "#89A1AD",
  header_text_col = "black",
  highlight = NULL,
  highlight_darken = 0.3,
  font_family = "Barlow",
  ...
)
```

Arguments

x	A table, typically a data.frame, tibble, or output from gtsummary.
font.size	The font size for text in the body of the table, defaults to 8 (passed through to set_flextable_defaults).
font.size.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 (passed through to set_flextable_defaults).
body_bg_col	Body background colour (default WAACHS cream).
header_bg_col	Header background colour (default WAACHS blue).
header_text_col	Header text colour (default black).
highlight	A numeric vector specifying rows to highlight.
highlight_darken	A numeric value specifying the amount by which body_bg_col should be "darkened" (tinted) (default 0.3).
font_family	Font family for plot (default Barlow).
...	Other arguments parsed to flextable::set_flextable_defaults.

Details

Inspired and based on thekidsbiostats::thekids_table().

Examples

```
head(mtcars) %>%
  waachs_table()
```

Index

∗ **dataset**
    icd_dat, 8

add_rows2, 2
aducust_flag, 3

check_att, 5
create_markdown, 6
create_project, 7, *7*
create_template, *6*

icd_dat, 8
icd_morb_flag, 9

person_summary, 12
proc_contents, 13
proc_freq, 14

save_waachs, 15
sum_tab, 15
sumfun, 16

theme_waachs, 16
transpose_tblsum, 17
twoway, 18

val_filt, 19

waachs_palette, 19
waachs_table, 20