Package ‘tabxplor’

March 8, 2024

Title  User-Friendly Tables with Color Helpers for Data Exploration

Version  1.1.3

Description  Make it easy to deal with multiple cross-tables in data
exploration, by creating them, manipulating them, and adding color
helpers to highlight important informations (differences from totals,
comparisons between lines or columns, contributions to variance, margins
of error, etc.). All functions are pipe-friendly and render data frames
which can be easily manipulated. In the same time, time-taking operations
are done with 'data.table' to go faster with big dataframes. Tables can
be exported to 'Excel' and in html with formats and colors.

URL  https://github.com/BriceNocenti/tabxplor

BugReports  https://github.com/BriceNocenti/tabxplor/issues

License  GPL (>= 3)

Encoding  UTF-8

RoxygenNote  7.3.1

Suggests  fansi (>= 0.5.0), htmltools (>= 0.5.0), knitr, openxlsx (>=
  4.0.0), rmarkdown, rstudioapi (>= 0.1), testthat (>= 3.0.0)

Config/testthat/edition  3

Imports  dplyr (>= 1.0.3), stringr (>= 1.4.0), crayon (>= 1.3.0),
  forcats (>= 0.5.0), magrittr (>= 1.5.0), purrr (>= 0.3.0),
  rlang (>= 0.4.0), tibble (>= 3.1.0), tidyr (>= 1.1.0), vctrs
  (>= 0.3.0), cli (>= 2.0.0), tidyselect (>= 1.0.0), stringi (>=
  1.4.6), pillar (>= 1.6.0), stats (>= 4.0.0), kableExtra (>=
  1.3.0), DescTools (>= 0.99.0), data.table

VignetteBuilder  knitr

NeedsCompilation  no

Author  Brice Nocenti [aut, cre]

Maintainer  Brice Nocenti <brice.nocenti@gmail.com>

Repository  CRAN

Date/Publication  2024-03-08 22:30:03 UTC
## R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>complete_partial_totals</td>
<td>4</td>
</tr>
<tr>
<td>dplyr_col_modify.tabxplor_grouped_tab</td>
<td>4</td>
</tr>
<tr>
<td>dplyr_reconstruct.tabxplor_grouped_tab</td>
<td>5</td>
</tr>
<tr>
<td>dplyr_row_slice.tabxplor_grouped_tab</td>
<td>5</td>
</tr>
<tr>
<td>fct_recode_helper</td>
<td>6</td>
</tr>
<tr>
<td>fmt</td>
<td>6</td>
</tr>
<tr>
<td>fmt_get_color_code</td>
<td>13</td>
</tr>
<tr>
<td>format.tabxplor_fmt</td>
<td>14</td>
</tr>
<tr>
<td>get_ci_type.data.frame</td>
<td>14</td>
</tr>
<tr>
<td>get_ci_type.default</td>
<td>15</td>
</tr>
<tr>
<td>get_ci_type.tabxplor_fmt</td>
<td>15</td>
</tr>
<tr>
<td>get_color.data.frame</td>
<td>16</td>
</tr>
<tr>
<td>get_color.default</td>
<td>16</td>
</tr>
<tr>
<td>get_color.tabxplor_fmt</td>
<td>17</td>
</tr>
<tr>
<td>get_col_var.data.frame</td>
<td>17</td>
</tr>
<tr>
<td>get_col_var.default</td>
<td>18</td>
</tr>
<tr>
<td>get_col_var.tabxplor_fmt</td>
<td>18</td>
</tr>
<tr>
<td>get_diff_type.data.frame</td>
<td>19</td>
</tr>
<tr>
<td>get_diff_type.default</td>
<td>19</td>
</tr>
<tr>
<td>get_diff_type.tabxplor_fmt</td>
<td>20</td>
</tr>
<tr>
<td>get_type.data.frame</td>
<td>20</td>
</tr>
<tr>
<td>get_type.default</td>
<td>21</td>
</tr>
<tr>
<td>get_type.tabxplor_fmt</td>
<td>21</td>
</tr>
<tr>
<td>group_by.tabxplor_tab</td>
<td>22</td>
</tr>
<tr>
<td>is_refcol.data.frame</td>
<td>22</td>
</tr>
<tr>
<td>is_refcol.default</td>
<td>23</td>
</tr>
<tr>
<td>is_refcol.tabxplor_fmt</td>
<td>23</td>
</tr>
<tr>
<td>is_refrow.data.frame</td>
<td>24</td>
</tr>
<tr>
<td>is_refrow.default</td>
<td>24</td>
</tr>
<tr>
<td>is_refrow.tabxplor_fmt</td>
<td>25</td>
</tr>
<tr>
<td>is_totcol.data.frame</td>
<td>25</td>
</tr>
<tr>
<td>is_totcol.default</td>
<td>26</td>
</tr>
<tr>
<td>is_totcol.tabxplor_fmt</td>
<td>26</td>
</tr>
<tr>
<td>is_totrow.data.frame</td>
<td>27</td>
</tr>
<tr>
<td>is_totrow.default</td>
<td>27</td>
</tr>
<tr>
<td>is_totrow.tabxplor_fmt</td>
<td>28</td>
</tr>
<tr>
<td>is_tottab.data.frame</td>
<td>28</td>
</tr>
<tr>
<td>is_tottab.default</td>
<td>29</td>
</tr>
<tr>
<td>is_tottab.tabxplor_fmt</td>
<td>29</td>
</tr>
<tr>
<td>mutate.tabxplor_fmt</td>
<td>30</td>
</tr>
<tr>
<td>new_tab</td>
<td>30</td>
</tr>
<tr>
<td>pillar_shaft.tabxplor_fmt</td>
<td>31</td>
</tr>
<tr>
<td>pillar_shaft.tab_chi2_fmt</td>
<td>32</td>
</tr>
<tr>
<td>print.tabxplor_grouped_tab</td>
<td>32</td>
</tr>
<tr>
<td>print.tabxplor_tab</td>
<td>33</td>
</tr>
<tr>
<td>relocate.tabxplor_grouped_tab</td>
<td>34</td>
</tr>
</tbody>
</table>
rename.tabxplor_grouped_tab .................................................. 34
rename_with.tabxplor_grouped_tab ............................................. 35
rowwise.tabxplor_grouped_tab .................................................. 35
rowwise.tabxplor_tab ............................................................. 36
select.tabxplor_grouped_tab .................................................... 36
summarise.tabxplor_grouped_tab ............................................... 37
tab ..................................................................................... 37
tab.chi2 ................................................................................. 42
tab.ci ..................................................................................... 43
tab.kable ............................................................................... 44
tab_many ............................................................................... 45
tab_num ............................................................................... 52
tab_pct ................................................................................... 54
tab_plain ............................................................................... 55
tab_prepare ............................................................................. 58
tab.spread .............................................................................. 59
tab_tot ..................................................................................... 60
tab.totaltab ........................................................................... 61
tab.xl ...................................................................................... 62
tab.xl_confidential ................................................................. 63
tbl_format_body.tabxplor_tab ..................................................... 65
tbl_format_footer.tabxplor_tab .................................................. 66
tbl_sum.tabxplor_grouped_tab ................................................... 66
tbl_sum.tabxplor_tab ............................................................... 67
ungroup.tabxplor_grouped_tab .................................................... 67
vec_arith.tabxplor_fmt ............................................................... 68
vec_cast.character.tabxplor_fmt .................................................. 69
vec_cast.double.tabxplor_fmt ...................................................... 70
vec_cast.integer.tabxplor_fmt ..................................................... 70
vec_cast.tabxplor_fmt.double ..................................................... 71
vec_cast.tabxplor_fmt.integer .................................................... 71
vec_cast.tabxplor_fmt.tabxplor_fmt .......................................... 72
vec_math.tabxplor_fmt .............................................................. 72
vec_proxy_compare.tabxplor_fmt ................................................ 73
vec_proxy_equal.tabxplor_fmt ...................................................... 73
vec_ptype2.double.tabxplor_fmt .................................................. 74
vec_ptype2.integer.tabxplor_fmt ................................................ 74
vec_ptype2.tabxplor_fmt.double ................................................ 75
vec_ptype2.tabxplor_fmt.integer ............................................... 75
vec_ptype2.tabxplor_fmt.tabxplor_fmt ...................................... 76
vec_ptype_abbr.tabxplor_fmt ...................................................... 76
vec_ptype_full.tabxplor_fmt ....................................................... 76
\[.tabxplor_grouped_tab ........................................................... 77
[<-.tabxplor_grouped_tab .......................................................... 78
[<-.tabxplor_grouped_tab .......................................................... 78
$.tabxplor_fmt ................................................................. 79

Index ................................................................. 80
complete_partial_totals

*Complete partial total rows*

**Description**

Complete partial total rows

**Usage**

`complete_partial_totals(tabs)`

**Arguments**

`tabs`  
A table or data.frame containing `tabxplor_fmt` columns.

**Value**

The table with completed total rows, total tables, and reference rows.

dplyr_col_modify.tabxplor_grouped_tab

dplyr_col_modify method for class `tabxplor_grouped_tab`

**Description**

dplyr_col_modify method for class `tabxplor_grouped_tab`

**Usage**

```r
## S3 method for class 'tabxplor_grouped_tab'
dplyr_col_modify(data, cols)
```

**Arguments**

`data`  
A data frame.

`cols`  
A named list used modify columns. A NULL value should remove an existing column.

**Value**

An object of class `tabxplor_grouped_tab`. 
### dplyr_reconstruct method for class tabxplor_grouped_tab

**Description**

Dplyr_reconstruct method for class tabxplor_grouped_tab

**Usage**

```r
## S3 method for class 'tabxplor_grouped_tab'
dplyr_reconstruct(data, template)
```

**Arguments**

- `data`: A data frame.
- `template`: Template to use for restoring attributes

**Value**

An object of class tabxplor_grouped_tab.

### dplyr_row_slice method for class tabxplor_grouped_tab

**Description**

Dplyr_row_slice method for class tabxplor_grouped_tab

**Usage**

```r
## S3 method for class 'tabxplor_grouped_tab'
dplyr_row_slice(data, i, ...)  
```

**Arguments**

- `data`: A data frame.
- `i`: A numeric or logical vector that indexes the rows of `.data`.
- `...`: Future parameters

**Value**

An object of class tabxplor_grouped_tab.
fct_recode_helper

fct_recode helper to recode multiple variables

Description

fct_recode helper to recode multiple variables

Usage

fct_recode_helper(
  .data,
  .cols = ~where(is.numeric),
  .data_out_name,
  cat = TRUE
)

Arguments

- `.data` The data frame.
- `.cols` `<tidy-select>` The variables to recode.
- `.data_out_name` The name of the output data frame, if different from the input data frame.
- `cat` By default the result is written in the console if there are less than 6 variables, written in a temporary file and opened otherwise. Set to false to get a data frame with a character variable instead.

Value

When the number of variables is less than 5, a text in console as a side effect. With more than 5 variables, a temporary R file. A tibble with the recode text as a character variable is returned invisibly (or as main result if `cat = TRUE`).

fmt

Create a vector of class formatted numbers

Description

fmt vectors, of class `tabxplor_fmt`, powers `tabxplor` and `tab` tibbles. As a record, they store all data necessary to calculate percentages, Chi2 metadata or confidence intervals, but also to format and color the table to help the user read it. You can access this data with `vctrs::field`, or change it with `vctrs::field<-`. A fmt vector have 13 fields: `n`, `digits`, `display`, `wn`, `pct`, `mean`, `diff`, `ctr`, `var`, `ci`, `in_totrow`, `in_tottab`, `in_refrow`. Other arguments are attributes, attached not to each value, but to the whole vector, like `type`, `totcol` or `color`. You can get them with `attr` and modify them with `attr<-`. Special functions listed below are made to facilitate programming with with `tabxplor` formatted numbers. taxplfmt vectors can use all standard operations, like `+`, `-`, `sum()`, or `c()`, using `vctrs`
Usage

```r
fmt(
    n = integer(),
    type = "n",
    digits = rep(0L, length(n)),
    display = dplyr::case_when(type == "mean" ~ "mean", type %in% c("row", "col", "all", "all_tabs") ~ "pct", TRUE ~ "n"),
    wn = rep(NA_real_, length(n)),
    pct = rep(NA_real_, length(n)),
    mean = rep(NA_real_, length(n)),
    diff = rep(NA_real_, length(n)),
    ctr = rep(NA_real_, length(n)),
    var = rep(NA_real_, length(n)),
    ci = rep(NA_real_, length(n)),
    in_totrow = rep(FALSE, length(n)),
    in_tottab = rep(FALSE, length(n)),
    in_refrow = rep(TRUE, length(n)),
    comp_all = NA,
    diff_type = "",
    ci_type = "",
    col_var = "",
    totcol = FALSE,
    refcol = FALSE,
    color = ""
)

is_fmt(x)
get_num(x)
set_num(x, value)
get_type(x, ...)
set_type(x, type)
is_totrow(x, ...)
as_totrow(x, in_totrow = TRUE)
is_tottab(x, ...)
as_tottab(x, in_tottab = TRUE)
is_totcol(x, ...)
as_totcol(x, totcol = TRUE)
```
is_refrow(x, ...)

as_refrow(x, in_refrow = TRUE)

get_comp_all(x, replace_na = TRUE)

set_comp_all(x, comp_all = FALSE)

get_diff_type(x, ...)

set_diff_type(x, diff_type)

get_ci_type(x, ...)

set_ci_type(x, ci_type)

get_col_var(x, ...)

set_col_var(x, col_var)

is_refcol(x, ...)

as_refcol(x, refcol = TRUE)

get_color(x, ...)

set_color(x, color)

get_digits(x)

set_digits(x, value)

Arguments

n The underlying count, as an integer vector of length n(). It is used to calculate confidence intervals.

type The type of the column, which defines the type of background calculation to be made (as a single string, since it's not a field but an attribute):

- "n": counts
- "mean": mean column (from numeric variables)
- "row": row percentages
- "col": column percentages
- "all": frequencies by subtable/group (i.e. by tab_vars)
- "all_tabs": frequencies for the whole table

digits The number of digits, as an integer, or an integer vector the length of n.

display The display type: the name of the field you want to show when printing the vector. Among "n", "wn", "pct", "diff", "ctr", "mean", "var", "ci", "pct_ci"
(percentages with visible confidence interval), "mean_ci" (means with visible confidence interval). As a single string, or a character vector the length of n.

*wn* The underlying weighted counts, as a double vector the length of n. It is used in certain operations on *fmt*, like means.

*pct* The percentages, as a double vector the length of n. Calculate with *tab_pct*.

*mean* The means, as a double vector the length of n.

*diff* The differences (from totals or first cells), as a double vector the length of n. Used to set colors for means and row or col percentages. Calculate with *tab_pct*.

*ctr* The contributions of cells to (sub)tables variances, as a double vector the length of n. Used to print colors when *color = "contrib"*. The mean contribution of each (sub)table is written on total rows (then, colors don’t print well without total rows). Calculate with *tab_chi2*.

*var* The cells variances, as a double vector the length of n. Used with *type = "mean"* to calculate confidence intervals. Calculate with *tab_plain*.

*ci* The confidence intervals, as a double vector the length of n. Used to print colors ("diff_ci", "after_ci"). Calculate with *tab_ci*.

*in_totrow* TRUE when the cell is part of a total row

*in_tottab* TRUE when the cell is part of a total table

*in_refrow* TRUE when the cell is part of a reference row (cf. *diff_type*)

*comp_all* FALSE when the comparison level is the subtable/group, TRUE when it is the whole table

*diff_type* The type of difference of the vector (calculate with *tab_pct*):

- "" or "no": no differences have been calculated
- "tot": the reference row (or column) is the total row (or column)
- "first": the reference row (or column) is the first row (or column)

*ci_type* The type of confidence intervals of the vector (calculate with *tab_ci*):

- "" or "no": no ci have been calculated
- "cell": absolute confidence intervals of cells percentages.
- "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when *diff_type = "first"*).
- "auto": "diff" for means and row/col percentages, "cell" for frequencies ("all", "all_tabs").

*col_var* The name of the *col_var* used to calculate the vector

*totcol* TRUE when the vector is a total column

*refcol* TRUE when the vector is a reference column

*color* The type of color to print:

- "no": no colors are printed.
- "diff": color percentages and means based on cells differences from totals (or from first cells when *diff = "first"*).
• "diff_ci": color pct and means based on cells differences from totals or first cells, removing coloring when the confidence interval of this difference is higher than the difference itself.
• "after_ci": idem, but cut off the confidence interval from the difference first.
• "contrib": color cells based on their contribution to variance (except mean columns, from numeric variables).

\( \text{x} \)

The object to test, to get a field in, or to modify.

\( \text{value} \)

The value you want to inject in some \texttt{fmt} vector's vctrs::field or attribute using a given "set" function.

\( \ldots \)

Used in methods to add arguments in the future.

\( \text{replace_na} \)

By default, \texttt{get_comp_all} takes NA in comparison level to be a FALSE (=comparison at subtables/groups level). Set to FALSE to avoid this behavior.

\textbf{Value}

A vector of class \texttt{tabxplor_fmt}.
A logical vector.
A double vector.
A \texttt{fmt} vector.
A character vector with the vectors type.
A \texttt{fmt} vector.
A logical vector with the \texttt{fmt} vectors totrow field.
A \texttt{fmt} vector with totrow field changed.
A logical vector with the \texttt{fmt} vectors tottab field.
A \texttt{fmt} vector with tottab field changed.
A logical vector with the \texttt{fmt} vectors totcol attribute.
A \texttt{fmt} vector with totcol attribute changed.
A logical vector with the \texttt{fmt} vectors in_refrow field
A \texttt{fmt} vector with in_refrow field changed.
A \texttt{fmt} vector with comp attribute changed.
A logical vector with the \texttt{fmt} vectors type attributes
A \texttt{fmt} vector.
A logical vector with the \texttt{fmt} vectors ci_type attributes
A \texttt{fmt} vector.
A logical vector with the \texttt{fmt} vectors col_var attributes
A \texttt{fmt} vector.
A logical vector with the \texttt{fmt} vectors is_refcol attributes
A \texttt{fmt} vector.
A logical vector with the \texttt{fmt} vectors color attributes
A \texttt{fmt} vector.
Functions

- **is_fmt()**: a test function for class fmt.
- **get_num()**: get the currently displayed field
- **set_num()**: set the currently displayed field (not changing display type)
- **get_type()**: get types of fmt columns (at fmt level or tab level)
- **set_type()**: set the column type attribute of a fmt vector
- **is_totrow()**: test function to detect cells in total rows (at fmt level or tab level)
- **as_totrow()**: set the "in_totrow" field (belong to total row)
- **is_tottab()**: test function to detect cells in total tables (at fmt level or tab level)
- **as_tottab()**: set the "in_tottab" field (belong to total table)
- **is_totcol()**: test function for total columns (at fmt level or tab level)
- **as_totcol()**: set the "totcol" attribute of a fmt vector
- **is_refrow()**: test function to detect cells in reference rows (at fmt level or tab level)
- **as_refrow()**: set the "in_refrow" field (belong to reference row)
- **get_comp_all()**: get comparison level of fmt columns
- **set_comp_all()**: set the comparison level attribute of a fmt vector
- **get_diff_type()**: get differences type of fmt columns (at fmt level or tab level)
- **set_diff_type()**: set the differences type attribute of a fmt vector
- **get_ci_type()**: get confidence intervals type of fmt columns (at fmt level or tab level)
- **set_ci_type()**: set the confidence intervals type attribute of a fmt vector
- **get_col_var()**: get names of column variable of fmt columns (at fmt level or tab level)
- **set_col_var()**: set the "col_var" attribute of a fmt vector
- **is_refcol()**: test function for reference columns (at fmt level or tab level)
- **as_refcol()**: set the "ref_col" attribute of a fmt vector
- **get_color()**: get color (at fmt level or tab level)
- **set_color()**: set the "color" attribute of a fmt vector
- **get_digits()**: get the "digits" field
- **set_digits()**: set the "digits" field

Examples

library(dplyr)

f <- fmt(n = c(7, 19, 2), type = "row", pct = c(0.25, 0.679, 0.07))
f

# To get the currently displayed field :
get_num(f)

# To modify the currently displayed field :
set_num(f, c(1, 0, 0))
# See all the underlying fields of a fmt vector (a data frame with a number of rows equal to the length of the vector):

```r
vctrs::vec_data(f)
```

# To get the numbers of digits:

```r
vctrs::field(f, "digits")
f$digits
```

# To get the count:

```r
vctrs::field(f, "n")
f$n
```

# To get the display:

```r
vctrs::field(f, "display")
f$display
```

# To modify a field, you can use `dplyr::mutate` on the fmt vector, referring to the names of the columns of the underlying data.frame (`vctrs::vec_data`):

```r
vctrs::field<- (f, "pct", c(1, 0, 0))
mutable(f, pct = c(1, 0, 0))
```

# See all the attributes of a fmt vector:

```r
attributes(f)
```

# To modify the "type" attribute of a fmt vector:

```r
set_type(f, "col")
```

# To modify the "color" attribute of a fmt vector:

```r
set_color(f, "contrib")
```

tabs <- tab(starwars, sex, hair_color, gender, na = "drop", pct = "row", rare_to_other = TRUE, n_min = 5)

# To identify the total columns, and work with them:

```r
is_totcol(tabs)
tabs %>% mutate(across(where(is_totcol), ~ "total column"))
```

# To identify the total rows, and work with them:

```r
is_totrow(tabs)
tabs %>% mutate(across(where(is_fmt), ~ if_else(is_totrow(.), true = "into_total_row", false = "normal_cell"))
```

# To identify the total tables, and work with them:

```r
tottabs <- is_tottab(tabs)
tabs %>% tibble::add_column(tottabs) %>%
mutate(total = if_else(tottabs, "part of a total table", "normal cell"))

# To access the displayed numbers, as numeric vectors :
tabs %>% mutate(across(where(is_fmt), get_num))

# To access the displayed numbers, as character vectors (without colors) :
tabs %>% mutate(across(where(is_fmt), format))

# To access the (non-displayed) differences of the cells percentages from totals :
tabs %>% mutate(across(where(is_fmt), ~ vctrs::field(., "diff")))

# To do more complex operations, like creating a new column with standard deviation and
# print it with 2 decimals, use `dplyr::mutate` on all the fmt columns of a table :
tab_num(forcats::gss_cat, race, c(age, tvhours), marital, digits = 1L, comp = "all") |>  
dplyr::mutate(dplyr::across( #Mutate over the whole table.  
c(age, tvhours),  
~ dplyr::mutate(.,
            var = sqrt(var),
            display = "var",
            digits = 2L) |>
        set_color("no"),
        .names = "{.col}_sd"
))

fmt_get_color_code

Get HTML Color Code of a fmt vector

Description

Get HTML Color Code of a fmt vector

Usage

fmt_get_color_code(x, type = "text", theme = "light", html_24_bit = NULL)

Arguments

x            The fmt vector to get the html color codes from.
type         The style type in set_color_style and get_color_style, "text" to color the
             text, "bg" to color the background.
theme        For set_color_style and get_color_style, is your console or html table
             background "light" or "dark"? Default to RStudio theme.
html_24_bit  Should specific 24bits colors palettes be used for html tables? With light themes
             only. Default to getOption("tabxplor.color_html_24_bit")
Value

A character vector with html color codes, of the length of the initial vector.

Examples

tabs <- tab(forcats::gss_cat, race, marital, pct = "row", color = "diff")
dplyr::mutate(tabs, across(where(is_fmt), fmt_get_color_code))

format.tabxplor_fmt  Print method for class tabxplor_fmt

Description

Print method for class tabxplor_fmt

Usage

## S3 method for class 'tabxplor_fmt'
format(x, ..., html = FALSE, na = NA)

Arguments

x  A fmt object.
...
Other parameters.
html  Should html tags be added (to print confidence intervals as subscripts) ?
na  How NAs should be printed. Default to NA.

Value

The fmt printed in a character vector.

get_ci_type.data.frame  Get confidence intervals type of fmt columns

Description

Get confidence intervals type of fmt columns

Usage

## S3 method for class 'data.frame'
get_ci_type(x, ...)

get_ci_type.data.frame  Get confidence intervals type of fmt columns
**get_ci_type.default**

**Arguments**

- **x**
  The object to test, to get a field in, or to modify.
- **...**
  Used in methods to add arguments in the future.

**Value**

A character vector with the ci_type attributes.

---

**get_ci_type.default** *Get confidence intervals type of fmt columns*

**Description**

Get confidence intervals type of fmt columns

**Usage**

```r
## Default S3 method:
get_ci_type(x, ...)
```

**Arguments**

- **x**
  The object to test, to get a field in, or to modify.
- **...**
  Used in methods to add arguments in the future.

**Value**

A single character with the ci_type attribute.

---

**get_ci_type.tabxplor_fmt** *Get confidence intervals type of fmt columns*

**Description**

Get confidence intervals type of fmt columns

**Usage**

```r
## S3 method for class 'tabxplor_fmt'
get_ci_type(x, ...)
```

**Arguments**

- **x**
  The object to test, to get a field in, or to modify.
- **...**
  Used in methods to add arguments in the future.
**get_color.default**

**Value**

A single character with the ci_type attribute.

**get_color.data.frame**  
*Get color*

**Description**

Get color

**Usage**

```r
## S3 method for class 'data.frame'
get_color(x, ...)
```

**Arguments**

- `x`  
The object to test, to get a field in, or to modify.
- `...`  
Used in methods to add arguments in the future.

**Value**

A character vector with the color attributes.

**get_color.default**  
*Get color*

**Description**

Get color

**Usage**

```r
## Default S3 method:
get_color(x, ...)
```

**Arguments**

- `x`  
The object to test, to get a field in, or to modify.
- `...`  
Used in methods to add arguments in the future.

**Value**

A single character with the color attribute.
### get_color.tabxplor_fmt

**Get color**

#### Description
Get color

#### Usage
```r
## S3 method for class 'tabxplor_fmt'
geget_color(x, ...)
```

#### Arguments
- `x` The object to test, to get a field in, or to modify.
- `...` Used in methods to add arguments in the future.

#### Value
A single character with the color attribute.

---

### get_col_var.data.frame

**Get names of column variable of fmt columns**

#### Description
Get names of column variable of fmt columns

#### Usage
```r
## S3 method for class 'data.frame'
geget_col_var(x, ...)
```

#### Arguments
- `x` The object to test, to get a field in, or to modify.
- `...` Used in methods to add arguments in the future.

#### Value
A character vector with the col_var attributes.
get_col_var.default  
Get names of column variable of fmt columns

Description
Get names of column variable of fmt columns

Usage
## Default S3 method:
get_col_var(x, ...)

Arguments
x  The object to test, to get a field in, or to modify.
... Used in methods to add arguments in the future.

Value
A single character with the col_var attribute.

get_col_var.tabxplor_fmt

Description
Get names of column variable of fmt columns

Usage
## S3 method for class 'tabxplor_fmt'
get_col_var(x, ...)

Arguments
x  The object to test, to get a field in, or to modify.
... Used in methods to add arguments in the future.

Value
A single character with the col_var attribute.
get_diff_type.data.frame

Get differences type of fmt columns

Description

Get differences type of fmt columns

Usage

## S3 method for class 'data.frame'
get_diff_type(x, ...)

Arguments

x The object to test, to get a field in, or to modify.
...
Used in methods to add arguments in the future.

Value

A character vector with the diff_type attribute.

get_diff_type.default

Get differences type of fmt columns

Description

Get differences type of fmt columns

Usage

## Default S3 method:
get_diff_type(x, ...)

Arguments

x The object to test, to get a field in, or to modify.
...
Used in methods to add arguments in the future.

Value

A single character with the diff_type attribute.
get_diff_type.tabxplor_fmt

Get differences type of fmt columns

Description

Get differences type of fmt columns

Usage

```r
## S3 method for class 'tabxplor_fmt'
get_diff_type(x, ...)
```

Arguments

- `x` The object to test, to get a field in, or to modify.
- `...` Used in methods to add arguments in the future.

Value

A single character with the diff_type attribute.

get_type.data.frame

Get types of fmt columns

Description

Get types of fmt columns

Usage

```r
## S3 method for class 'data.frame'
get_type(x, ...)
```

Arguments

- `x` The object to test, to get a field in, or to modify.
- `...` Used in methods to add arguments in the future.

Value

A character vector with the data.frame column’s types.
get_type.default

### Description
Get types of fmt columns

### Usage
```r
## Default S3 method:
get_type(x, ...)
```

### Arguments
- `x`: The object to test, to get a field in, or to modify.
- `...`: Used in methods to add arguments in the future.

### Value
An empty character vector.

get_type.tabxplor_fmt

### Description
Get types of fmt columns

### Usage
```r
## S3 method for class 'tabxplor_fmt'
get_type(x, ...)
```

### Arguments
- `x`: The object to test, to get a field in, or to modify.
- `...`: Used in methods to add arguments in the future.

### Value
A single string with the vector’s type.
### `group_by.tabxplor_tab`  
**group_by method for class `tabxplor_tab`**

**Description**

`group_by` method for class `tabxplor_tab`

**Usage**

```r
## S3 method for class 'tabxplor_tab'
group_by(.data, ..., .add = FALSE, .drop = dplyr::group_by_drop_default(.data))
```

**Arguments**

- `.data`  
  A tibble of class `tabxplor_tab`.

- `...`  
  Variables or computations to group by.

- `.add`  
  When `FALSE`, the default, `group_by()` will override existing groups. To add to the existing groups, use `.add = TRUE`.

- `.drop`  
  Drop groups formed by factor levels that don’t appear in the data? The default is `TRUE` except when `.data` has been previously grouped with `.drop = FALSE`.

**Value**

A tibble of class `tabxplor_grouped_tab`.

---

### `is_refcol.data.frame`  
**Test function for reference columns**

**Description**

Test function for reference columns

**Usage**

```r
## S3 method for class 'data.frame'
is_refcol(x, ...)
```

**Arguments**

- `x`  
  The object to test, to get a field in, or to modify.

- `...`  
  Used in methods to add arguments in the future.

**Value**

A character vector with the `ref_col` attributes.
**is_refcol.default**

*Test function for reference columns*

**Description**

Test function for reference columns

**Usage**

```r
## Default S3 method:
is_refcol(x, ...)
```

**Arguments**

- `x`  
  The object to test, to get a field in, or to modify.

- `...`  
  Used in methods to add arguments in the future.

**Value**

A single character with the ref_col attribute.

---

**is_refcol.tabxplor_fmt**

*Test function for reference columns*

**Description**

Test function for reference columns

**Usage**

```r
## S3 method for class 'tabxplor_fmt'
is_refcol(x, ...)
```

**Arguments**

- `x`  
  The object to test, to get a field in, or to modify.

- `...`  
  Used in methods to add arguments in the future.

**Value**

A single character with the ref_col attribute.
is_refrow.data.frame  Test function to detect cells in reference rows

Description
Test function to detect cells in reference rows

Usage
## S3 method for class 'data.frame'
is_refrow(x, ..., partial = TRUE)

Arguments
x The object to test, to get a field in, or to modify.
... Used in methods to add arguments in the future.
partial Should partial reference rows be counted as reference rows? Default to FALSE.

Value
A list of logical vectors with the in_refrow fields.

is_refrow.default  Test function to detect cells in reference rows

Description
Test function to detect cells in reference rows

Usage
## Default S3 method:
is_refrow(x, ...)

Arguments
x The object to test, to get a field in, or to modify.
... Used in methods to add arguments in the future.

Value
A logical vector with FALSE, the length of x.
is_refrow.tabxplor_fmt

Test function to detect cells in reference rows

Description

Test function to detect cells in reference rows

Usage

```r
## S3 method for class 'tabxplor_fmt'
is_refrow(x, ...)
```

Arguments

- `x`: The object to test, to get a field in, or to modify.
- `...`: Used in methods to add arguments in the future.

Value

A logical vector with the in_refrow field.

is_totcol.data.frame  Test function for total columns

Description

Test function for total columns

Usage

```r
## S3 method for class 'data.frame'
is_totcol(x, ...)
```

Arguments

- `x`: The object to test, to get a field in, or to modify.
- `...`: Used in methods to add arguments in the future.

Value

A logical vector, with the data.frame column’s totcol attributes.
is_totcol.default  Test function for total columns

Description
Test function for total columns

Usage
## Default S3 method:
is_totcol(x, ...)

Arguments
x  The object to test, to get a field in, or to modify.
...  Used in methods to add arguments in the future.

Value
A single logical vector with the totcol attribute

is_totcol.tabxplor_fmt

Description
Test function for total columns

Usage
## S3 method for class 'tabxplor_fmt'
is_totcol(x, ...)

Arguments
x  The object to test, to get a field in, or to modify.
...  Used in methods to add arguments in the future.

Value
A single logical vector with the totcol attribute
is_totrow.data.frame Test function to detect cells in total rows

Description
Test function to detect cells in total rows

Usage
## S3 method for class 'data.frame'
is_totrow(x, ..., partial = FALSE)

Arguments
- \( x \) The object to test, to get a field in, or to modify.
- \( \ldots \) Used in methods to add arguments in the future.
- \( \text{partial} \) Should partial total rows be counted as total rows? Default to FALSE.

Value
A list of logical vectors, with the data.frame column’s totrow fields.

is_totrow.default Test function to detect cells in total rows

Description
Test function to detect cells in total rows

Usage
## Default S3 method:
is_totrow(x, \ldots)

Arguments
- \( x \) The object to test, to get a field in, or to modify.
- \( \ldots \) Used in methods to add arguments in the future.

Value
A logical vector with FALSE.
is_totrow.tabxplor_fmt

Test function to detect cells in total rows

Description
Test function to detect cells in total rows

Usage
## S3 method for class 'tabxplor_fmt'
is_totrow(x, ...)

Arguments
- x: The object to test, to get a field in, or to modify.
- ...: Used in methods to add arguments in the future.

Value
A logical vector with the totrow field.

is_tottab.data.frame
Test function to detect cells in total tables

Description
Test function to detect cells in total tables

Usage
## S3 method for class 'data.frame'
is_tottab(x, ..., partial = FALSE)

Arguments
- x: The object to test, to get a field in, or to modify.
- ...: Used in methods to add arguments in the future.
- partial: Should partial total tabs be counted as total tabs? Default to FALSE.

Value
A list of logical vectors, with the data.frame column’s tottab fields.
Description
Test function to detect cells in total tables

Usage
## Default S3 method:
is_tottab(x, ...)

Arguments
x The object to test, to get a field in, or to modify.
... Used in methods to add arguments in the future.

Value
A logical vector with FALSE.

--
is_tottab.tabxplor_fmt

Test function to detect cells in total tables

Description
Test function to detect cells in total tables

Usage
## S3 method for class 'tabxplor_fmt'
is_tottab(x, ...)

Arguments
x The object to test, to get a field in, or to modify.
... Used in methods to add arguments in the future.

Value
A logical vector with the tottab field.
**mutate.tabxplor_fmt**  
*mutate method to access vctrs::fields of tabxplor_fmt vectors*

**Description**

mutate method to access vctrs::fields of tabxplor_fmt vectors

**Usage**

```r
## S3 method for class 'tabxplor_fmt'
mutate(.data, ...)
```

**Arguments**

- `.data`  
  A `tabxplor_fmt` column.
- `...`  
  `<data-masking>` Name-value pairs. The name gives the name of the column in the output (do not change it).
  The value can be:
  - A vector of length 1, which will be recycled to the correct length.
  - A vector the same length as the current group (or the whole data frame if ungrouped).

**Value**

An object of class `tabxplor_fmt`.

---

**new_tab**  
*A constructor for class tabxplor_tab*

**Description**

A constructor for class `tabxplor_tab`

**Usage**

```r
new_tab(
  tabs = tibble::tibble(),
  subtext = "",
  chi2 = tibble::tibble(tables = character(), pvalue = double(), df = integer(), cells = integer(), variance = double(), count = integer()),
  ...
)
```

```r
new_grouped_tab(
  ..
)```
pillar_shaft.tabxplor_fmt

```r
tabs = tibble::tibble(),
groups,
subtext = "",
chi2 = tibble::tibble(tables = character(), pvalue = double(), df = integer(), cells = integer(), variance = double(), count = integer()),
...,
class = character()
)
```

**Arguments**

- **tabs**
  - A table, stored into a *tibble* data.frame. It is generally made with `tab`, `tab_many` or `tab_plain`.

- **subtext**
  - A character vector to print legend lines under the table.

- **chi2**
  - A tibble storing information about pvalues and variances, to fill with `tab_chi2`.

- **...**
  - Needed to implement subclasses.

- **class**
  - Needed to implement subclasses.

- **groups**
  - The grouping data.

**Value**

- A tibble of class `tabxplor_tab`.
- A tibble of class `tabxplor_grouped_tab`.

---

pillar_shaft.tabxplor_fmt

_Pillar_shaft method to print class fmt in a *tibble* column_

**Description**

Pillar_shaft method to print class fmt in a *tibble* column

**Usage**

```r
## S3 method for class 'tabxplor_fmt'
pillar_shaft(x, ...)
```

**Arguments**

- **x**
  - A fmt object.

- **...**
  - Other parameter.

**Value**

A fmt printed in a pillar.
pillar_shaft.tab_chi2_fmt

Print Chi2 tables columns

Description

Print Chi2 tables columns

Usage

## S3 method for class 'tab_chi2_fmt'
pillar_shaft(x, ...)

Arguments

x
A fmt object.

...  
Other parameter.

Value

A Chi2 table column printed in a pillar.

print.tabxplor_grouped_tab

Printing method for class tabxplor_grouped_tab

Description

Printing method for class tabxplor_grouped_tab

Usage

## S3 method for class 'tabxplor_grouped_tab'
print(
x,
width = NULL,
...,  
n = 100,
max_extra_cols = NULL,
max_footer_lines = NULL,
min_row_var = 30
)


print.tabxplor_tab

Arguments

- x: Object to format or print.
- width: Width of text output to generate.
- ...: Passed on to tbl_format_setup().
- n: Number of rows to show.
- max_extra_cols: Number of extra columns to print abbreviated information for, if the width is too small for the entire tibble.
- max_footer_lines: Maximum number of footer lines.
- min_row_var: Minimum number of characters for the row variable. Default to 30.

Value

A printed grouped table.

print.tabxplor_tab Printing method for class tabxplor_tab

Description

Printing method for class tabxplor_tab

Usage

## S3 method for class 'tabxplor_tab'
print(
  x,
  width = NULL,
  ..., 
  n = 100,
  max_extra_cols = NULL,
  max_footer_lines = NULL,
  min_row_var = 30
)

Arguments

- x: Object to format or print.
- width: Width of text output to generate.
- ...: Passed on to tbl_format_setup().
- n: Number of rows to show.
- max_extra_cols: Number of extra columns to print abbreviated information for, if the width is too small for the entire tibble.
- max_footer_lines: Maximum number of footer lines.
- min_row_var: Minimum number of characters for the row variable. Default to 30.
### relocate.tabxplor_grouped_tab

**Description**

relocate method for class tabxplor_grouped_tab

**Usage**

```r
## S3 method for class 'tabxplor_grouped_tab'
relocate(.data, ...)
```

**Arguments**

- `.data` A tibble of class tabxplor_tab.
- `...` Columns to move. will move columns to the left-hand side; specifying both is an error.

**Value**

An object of class tabxplor_grouped_tab.

---

### rename.tabxplor_grouped_tab

**Description**

rename method for class tabxplor_grouped_tab

**Usage**

```r
## S3 method for class 'tabxplor_grouped_tab'
rename(.data, ...)
```

**Arguments**

- `.data` A tibble of class tabxplor_tab.
- `...` Use `new_name = old_name` to rename selected variables.

**Value**

An object of class tabxplor_grouped_tab.
rename_with.tabxplor_grouped_tab

rename_with method for class tabxplor_grouped_tab

Description
rename_with method for class tabxplor_grouped_tab

Usage

## S3 method for class 'tabxplor_grouped_tab'
rename_with(.data, .fn, .cols = dplyr::everything(), ...)

Arguments
.data
A tibble of class tabxplor_tab.

.fn
A function used to transform the selected .cols. Should return a character vector the same length as the input.

.cols
Columns to rename; defaults to all columns.

... Additional arguments passed onto .fn.

Value
An object of class tabxplor_grouped_tab.

rowwise.tabxplor_grouped_tab

rowwise method for class tabxplor_grouped_tab

Description
rowwise method for class tabxplor_grouped_tab

Usage

## S3 method for class 'tabxplor_grouped_tab'
rowwise(data, ...)

Arguments
data
A tibble of class tabxplor_tab.

... Variables to be preserved when calling summarise(). This is typically a set of variables whose combination uniquely identify each row.

Value
An object of class tabxplor_grouped_tab and rowwise_df.
select.tabxplor_grouped_tab

Description

select method for class tabxplor_grouped_tab

Usage

## S3 method for class 'tabxplor_grouped_tab'
select(.data, ...)

Arguments

.data  A tibble of class tabxplor_grouped_tab.
...

Value

An object of class tabxplor_grouped_tab.

---

rowwise.tabxplor_tab  rowwise method for class tabxplor_tab

Description

rowwise method for class tabxplor_tab

Usage

## S3 method for class 'tabxplor_tab'
rowwise(data, ...)

Arguments

.data  A tibble of class tabxplor_tab.
...

Value

A tibble of class tabxplor_grouped_tab and rowwise_df.

---
summarise.tabxplor_grouped_tab

Description
summarise method for class tabxplor_grouped_tab

Usage
## S3 method for class 'tabxplor_grouped_tab'
summarise(.data, ..., .groups = NULL)

Arguments
.data A tibble of class tabxplor_tab.
... Name-value pairs of summary functions. The name will be the name of the variable in the result.
.groups Grouping structure of the result.

Value
An object of class tabxplor_grouped_tab.

tab

Single cross-table, with color helpers

Description
A full-featured function to create, manipulate and format single cross-tables, using colors to make the printed tab more easily readable (in R terminal or exported to Excel with tab_xl). Since objects of class tab are also of class tibble, you can then use all dplyr verbs to modify the result, like select, like arrange, filter or mutate. Wrapper around the more powerful tab_many.

Usage
tab(
data, row_var, col_var, tab_vars, wt, sup_cols, na = "keep", digits = 0,
pct = "no",
color = "no",
diff = "tot",
comp = "tab",
totaltab = "line",
totaltab_name = "Ensemble",
tot = c("row", "col"),
total_names = "Total",
chi2 = FALSE,
ci = "no",
conf_level = 0.95,
subtext = "",
cleannames = NULL,
rare_to_other = FALSE,
n_min = 30,
other_level = "Others",
filter
)

Arguments

data

A data frame.

row_var, col_var

The row variable, which will be printed with one level per line, and the column
variable, which will be printed with one level per column. For numeric variables
means are calculated, in a single column.

tab_vars

<Tidy-select> Tab variables : a subtable is made for each combination of levels of
the selected variables. Leave empty to make a simple cross-table. All tab_vars
are converted to factor.

wt

A weight variable, of class numeric. Leave empty for unweighted results.

sup_cols

<Tidy-select> Supplementary columns variables, with only the first level printed,
and row percentages (for numeric variables, a mean will be calculated for each
row_var). To pass many variables you may use syntax sup_cols = c(sup_col1,
sup_col2, ...). To keep all levels of other col_vars, or other types of per-
centages, use tab_many instead.

na

The policy to adopt for missing values, as a single string :

• "keep": by default, NA’s of row, col and tab variables are printed as an
  explicit "NA" level.
• "drop": remove NA’s in row, col and tab variables before calculations are
done. Supplementary columns are then calculated for observations with no
NA in any of the row, col and tab variables.

digits

The number of digits to print, as a single integer. To print a different number
of digits for each sup_cols, an integer vector of length 1 + sup_cols (the first
being the number of digits for the base table).

pct

The type of percentages to calculate :

• "row": row percentages.
- "col": column percentages.
- "all": frequencies for each subtable/group, if there is tab_vars.
- "all_tabs": frequencies for the whole (set of) table(s).

**color**
The type of colors to print, as a single string :
- "no": by default, no colors are printed.
- "diff": color percentages and means based on cells differences from totals (or from first cells when diff = "first").
- "diff_ci": color pct and means based on cells differences from totals or first cells, removing coloring when the confidence interval of this difference is higher than the difference itself.
- "after_ci": idem, but cut off the confidence interval from the difference first.
- "contrib": color cells based on their contribution to variance (except mean columns, from numeric variables).
- "auto": frequencies (pct = "all", pct = "all_tabs") and counts are colored with "contrib". When ci = "diff", row and col percentages are colored with "after_ci"; otherwise they are colored with "diff".

**diff**
The reference cell to calculate differences (used to print colors) :
- "tot": by default, cells differences from total rows are calculated with pct = "row", and cells differences from total columns with pct = "col".
- "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).
- "no": not use diffs to gain calculation time.

**comp**
The comparison level : by subtables/groups, or for the whole table.
- "tab": by default, contributions to variance, row differences from totals/first cells, and row confidence intervals for these differences, are calculated for each tab_vars group.
- "all": compare cells to the general total line (provided there is a total table with a total row), or with the first line of the total table when diff = "first".

**totaltab**
The total table, if there are subtables/groups (i.e. when tab_vars is provided) :
- "line": by default, add a general total line (necessary for calculations with comp = "all")
- "table": add a complete total table (i.e. row_var by col_vars without tab_vars).
- "no": not to draw any total table.

**totaltab_name**
The name of the total table, as a single string.

**tot**
The totals :
- c("col", "row") or "both": by default, both total rows and total columns.
- "row": only total rows.
- "col": only total column.
- "no": remove all totals (after calculations if needed).
The names of the totals, as a character vector of length one or two. Use syntax of type c("Total row", "Total column") to set different names for rows and cols.

chi2
Set to TRUE to calculate Chi2 summaries with tab.chi2. Useful to print metadata, and to color cells based on their contribution to variance (color = "contrib"). Automatically added if needed for color.

ci
The type of confidence intervals to calculate, passed to tab.ci (automatically added if needed for color).

• "cell": absolute confidence intervals of cells percentages.
• "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when diff = "first").
• "auto": ci = "diff" for means and row/col percentages, ci = "cell" for frequencies ("all", "all.tabs").

By default, for percentages, with ci = "cell" Wilson's method is used, and with ci = "diff" Wald's method along Agresti and Caffo's adjustment. Means use classic method. This can be changed in tab.ci.

conf_level
The confidence level, as a single numeric between 0 and 1. Default to 0.95 (95%).

subtext
A character vector to print rows of legend under the table.

cleannames
Set to TRUE to clean levels names, by removing prefix numbers like "1-", and text in parenthesis. All data formatting arguments are passed to tab.prepare.

rare_to_other
When set to TRUE, levels with less count than n.min will be merged into an "Other" level.

n.min
The count under which a level is aggregated in the "Other" level.

other_level
The name of the "Other" level, as a single string.

filter
A dplyr::filter to apply to the data frame first, as a single string (which will be converted to code, i.e. to a call). Useful when printing multiples tabs with tibble::tribble, to use different filters for similar tables or simply make the field of observation more visible into the code.

Value
A tibble of class tab, possibly with colored reading helpers. All non-text columns are of class fmt, storing all the data necessary to print formats and colors. Columns with row_var and tab_vars are of class factor: every added factor will be considered as a tab_vars and used for grouping. To add text columns without using them in calculations, be sure they are of class character.

Examples
# A simple cross-table:
tab(forcats::gss_cat, marital, race)

# With more variables provided, `tab` makes a subtables for each combination of levels:
tab(forcats::gss_cat, marital, tab_vars = c(year, race))
# You can also add supplementary columns, text or numeric:

```r
tab(dplyr::storms, category, status, sup_cols = c("pressure", "wind"))
```

# Colors to help the user read the table:

```r
data <- forcats::gss_cat %>%
  dplyr::filter(year %in% c(2000, 2006, 2012), marital %in% c("No answer", "Widowed"))
gss <- "Source: General social survey 2000-2014"
gss2 <- "Source: General social survey 2000, 2006 and 2012"

# Differences between the cell and it's subtable's total cell:

```r
tab(data, race, marital, year, subtext = gss2, pct = "row", color = "diff")
```

# Differences between the cell and the whole table's general total cell:

```r
tab(data, race, marital, year, subtext = gss2, pct = "row", color = "diff", comp = "all")
```

# Historical differences:

```r
data2 <- data %>% dplyr::mutate(year = as.factor(year))
tab(data2, year, marital, race, subtext = gss2, pct = "row", color = "diff", diff = "first", tot = "col")
```

# Differences with the total, except if their confidences intervals are superior to them:

```r
tab(forcats::gss_cat, race, marital, subtext = gss, pct = "row", color = "diff_ci")
```

# Same differences, minus their confidence intervals:

```r
tab(forcats::gss_cat, race, marital, subtext = gss, pct = "row", color = "after_ci")
```

# Contribution of cells to table's variance, like in a correspondence analysis:

```r
tab(forcats::gss_cat, race, marital, subtext = gss, color = "contrib")
```

# Since the result is a tibble, you can use all dplyr verbs to modify it:

```r
library(dplyr)
tab(dplyr::storms, category, status, sup_cols = c("pressure", "wind")) %>%
  dplyr::filter(category != "-1") %>%
  dplyr::select(-"tropical depression") %>%
  dplyr::arrange(is_totrow(.), desc(category))
```

# With `dplyr::arrange`, don't forget to keep the order of tab variables and total rows:

```r
tab(data, race, marital, year, pct = "row") %>%
```
Add Chi2 summaries to a `tab`

**Usage**

```r
dplyr::arrange(year, is_totrow(.), desc(Married))
```

**Description**

Add Chi2 summaries to a `tab`

**Arguments**

- `tabs` A tibble of class `tab`, made with `tab_plain` or `tab_many`.
- `calc` By default all elements of the Chi2 summary are calculated: contributions to variance, pvalue, variance and unweighted count. You can choose which are computed by selecting elements in the vector `c("ctr", "p", "var", "counts")`.
- `comp` Comparison level. When `tab_vars` are present, should the contributions to variance be calculated for each subtable/group (by default, `comp = "tab"`)? Should they be calculated for the whole table (`comp = "all"`)? `comp` must be set once and for all the first time you use `tab_plain`, `tab_num` or `tab_chi2` with rows, or `tab_ci`.
- `color` The type of colors to print, as a single string.
  - "no": by default, no colors are printed
  - "all": color all cells based on their contribution to variance (except for mean columns, from numeric variables)
  - "all_pct": color all percentages cells based on their contribution to variance
  - "auto": only color columns with counts, `pct = "all"` or `pct = "all_tabs"`

**Value**

A tibble of class `tab`, with Chi2 summaries as metadata, possibly colored based on contributions of cells to variance.
Add confidence intervals to a tab

**Description**

Add confidence intervals to a tab

**Usage**

```r
tab_ci(
  tabs,
  ci = "auto",
  comp = NULL,
  conf_level = 0.95,
  color = "no",
  visible = FALSE,
  method_cell = "wilson",
  method_diff = "ac"
)
```

**Arguments**

- **tabs**: A tibble of class tab made with `tab_plain` or `tab_many`.
- **ci**: The type of ci to calculate. Set to "cell" to calculate absolute confidence intervals. Set to "diff" to calculate the confidence intervals of the difference between a cell and the relative total cell (or the reference cell, when `diff` is not "tot" in `tab_plain` or `tab_num`). By default, "diff" ci are calculated for means and row and col percentages, "cell" ci for frequencies ("all", "all_tabs").
- **comp**: Comparison level. When `tab_vars` are present, should the contributions to variance be calculated for each subtable/group (by default, `comp = "tab"`) ? Should they be calculated for the whole table (`comp = "all"`) ? `comp` must be set once and for all the first time you use `tabPlain`, `tab_num` or `tab_ch2` with rows, or `tab_ci`.
- **conf_level**: The confidence level, as a single numeric between 0 and 1. Default to 0.95 (95%).
- **color**: The type of colors to print, as a single string.
  - "no": by default, no colors are printed
  - "diff_ci": color pct and means based on cells differences from totals or first cells, removing coloring when the confidence interval of this difference is higher than the difference itself
  - "after_ci": idem, but cut off the confidence interval from the difference
- **visible**: By default confidence intervals are calculated and used to set colors, but not printed. Set to TRUE to print them in the result.
method_cell Character string specifying which method to use with percentages for \( \text{ci} = \text{"cell"} \). This can be one out of: "wald", "wilson", "wilsoncc", "agresti-coull", "jeffreys", "modified wilson", "modified jeffreys", "clopper-pearson", "arcsine", "logit", "witting", "pratt", "midp", "lik" and "blaker". Defaults to "wilson". See \texttt{BinomCI}.

method_diff Character string specifying which method to use with percentages for \( \text{ci} = \text{"diff"} \). This can be one out of: "wald", "waldcc", "ac", "score", "scorecc", "mn", "mee", "blj", "ha", "hal", "jp". Defaults to "ac", Wald interval with the adjustment according to Agresti, Caffo for difference in proportions and independent samples. See \texttt{BinomDiffCI}.

Value

A \texttt{tibble} of class \texttt{tab}, colored based on differences (from totals/first cells) and confidence intervals.

Examples

# A typical workflow with tabxplor step-by-step functions :

```r
data <- dplyr::starwars %>%
  tab_prepare(sex, hair_color, gender, rare_to_other = TRUE, n_min = 5, na_drop_all = sex)
data %>%
  tab_plain(sex, hair_color, gender, tot = c("row", "col"), pct = "row", comp = "all") %>%
  tab_ci("diff", color = "after_ci")
```

---

**tab_kable**

*Print a tabxplor table in html*

**Description**

Print a tabxplor table in html

**Usage**

```r
tab_kable(
  tabs,
  theme = c("light", "dark"),
  color_type = NULL,
  html_24_bit = NULL,
  tooltips = TRUE,
  popover = NULL,
  color_legend = TRUE,
  caption = NULL,
  ...
)
```
**tab_many**

Many cross-tables as one, with color helpers

---

**Description**

A full-featured function to create, manipulate and format many cross-tables as one, using colors to make the printed tab more easily readable (in R terminal or exported to Excel with `tab_xl`). Since objects of class `tab` are also of class `tibble`, you can then use all `dplyr` verbs to modify the result, like `select`, `arrange`, `filter` or `mutate`.

Only breaks for attractions/over-representations (in green) should be given, as a vector of positive doubles, with length between 1 and 5. Breaks for aversions/under-representations (in orange/red) will simply be the opposite.

---

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tabs</code></td>
<td>A table made with <code>tab</code> or <code>tab_many</code>.</td>
</tr>
<tr>
<td><code>theme</code></td>
<td>By default, a white table with black text. Set to &quot;dark&quot; for a black table with white text.</td>
</tr>
<tr>
<td><code>color_type</code></td>
<td>Set to &quot;text&quot; to color the text, &quot;bg&quot; to color the background. By default it takes <code>getOption(&quot;tabxplor.color_style_type&quot;)</code>.</td>
</tr>
<tr>
<td><code>html_24_bit</code></td>
<td>Should specific 24bits colors palettes be used? Default to <code>getOption(&quot;tabxplor.color_html_24_bit&quot;)</code>.</td>
</tr>
<tr>
<td><code>tooltips</code></td>
<td>By default, html tooltips are used to display additional informations at mouse hover. Set to FALSE to discard.</td>
</tr>
<tr>
<td><code>popover</code></td>
<td>By default, takes <code>getOption(&quot;tabxplor.kable_popover&quot;)</code>. When FALSE, html tooltips are of the base kind: they can’t be used with floating table of content in rmarkdown documents. Set to TRUE to use <code>kableExtra</code> html popovers instead, which are compatible with floating toc. Remember to enable the popover module by copying the following code into your document: <code>&lt;script&gt; $(document).ready(function(){\$([data-toggle=&quot;popover&quot;]).popover();}); &lt;/script&gt;</code></td>
</tr>
<tr>
<td><code>color_legend</code></td>
<td>Print colors legend below the table? You can then use a css chunk in rmarkdown to change popovers colors.</td>
</tr>
<tr>
<td><code>caption</code></td>
<td>The table caption. For formatting, you need to use a css with caption{} in rmarkdown.</td>
</tr>
</tbody>
</table>

Other arguments to pass to `kableExtra::kable_styling`.

---

**Value**

A html table (opened in the viewer in RStudio). Differences from totals, confidence intervals, contribution to variance, and unweighted counts, are available in an html tooltip at cells hover.

---

**Examples**

```r
tabs <- tab(forcats::gss_cat, race, marital, year, pct = "row", color = "diff")
tab_kable(tabs, theme = "light", color_type = "text")
```

---

...
Usage

tab_many(
  data,
  row_vars,
  col_vars,
  tab_vars,
  wt,
  levels = "all",
  na = "keep",
  na_drop_all,
  digits = 0,
  totaltab = "line",
  totaltab_name = "Ensemble",
  totrow = TRUE,
  totcol = "last",
  total_names = "Total",
  pct = "no",
  diff = "tot",
  comp = "tab",
  chi2 = FALSE,
  ci = "no",
  conf_level = 0.95,
  method_cell = "wilson",
  method_diff = "ac",
  color = "no",
  subtext = "",
  cleannames = NULL,
  rare_to_other = FALSE,
  n_min = 30,
  other_level = "Others",
  filter
)

is_tab(x)

set_color_style(
  type = c("text", "bg"),
  theme = NULL,
  html_24_bit = c("blue_red", "green_red", "no"),
  custom_palette = NULL
)

get_color_style(
  mode = c("crayon", "color_code"),
  type = NULL,
  theme = NULL,
html_24_bit = NULL
)

set_color_breaks(pct_breaks, mean_breaks, contrib_breaks)

get_color_breaks(brk, type = c("positive", "all"))

Arguments

data
row_vars
col_vars
	<tidy-select> One column is printed for each level of each column variable. For numeric variables means are calculated, in a single column. To pass many variables you may use syntax col_vars = c(col_var1, col_var2, ...).
tab_vars
	<tidy-select> One subtable is made for each combination of levels of the tab variables. To pass many variables you may use syntax tab_vars = c(tab_var1, tab_var2, ...). All tab variables are converted to factor. Leave empty to make a simple table.
wt
levels
	The levels of col_vars to keep (for more complex selections use dplyr::select). The argument is vectorised over col_vars.
	• "all": by default, all levels are kept.
	• "first": only keep the first level of each col_vars
na
	The policy to adopt with missing values. It must be a single string.
	• na = "keep": by default, prints NA’s as explicit "NA" level.
	• na = "drop": removes NA levels before making each table (tabs made with different column variables may have a different number of observations, and won’t exactly have the same total columns).
na_drop_all
	<tidy-select> Removes all observations with a NA in any of the chosen variables, for all tables (tabs for each column variable will have the same number of observations).
digits

totaltab
	The total table, if there are subtables/groups (i.e. when tab_vars is provided). Vectorised over row_vars.
	• "line": by default, add a general total line (necessary for calculations with comp = "all")
	• "table": add a complete total table (i.e. row_var by col_vars without tab vars).
	• "no": not to draw any total table.
totaltab_name
	The name of the total table, as a single string.
**totrow**  By default, total rows are printed. Set to `FALSE` to remove them (after calculations if needed). Vectorised over `row_vars`.

**totcol**  The policy with total columns. Vectorised over `col_vars`.
- "last": by default, only prints a total column for the last column variable (of class factor, not numeric).
- "each": print a total column for each column variable.
- "no": remove all total columns (after calculations if needed).

**total_names**  The names of the totals, as a character vector of length one or two. Use syntax of type `c("Total row", "Total column")` to set different names for rows and cols.

**pct**  The type of percentages to calculate:
- "row": row percentages.
- "col": column percentages.
- "all": frequencies for each subtable/group, if there is `tab_vars`.
- "all_tabs": frequencies for the whole (set of) table(s).

The argument is vectorised over both `row_vars` and `col_vars`. You can then write as the following: `pct = list(row_var1 = list("row", "col", "col"), row_var2 = list("col", "row", "row"))`

**diff**  The reference cell to calculate differences (used to print colors). Vectorized over `row_vars`.
- "tot": by default, cells differences from total rows are calculated with `pct = "row"`, and cells differences from total columns with `pct = "col"`.
- "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).
- "n": when `diff` is an integer, the nth row (or column) is used for comparison.
- "regex": when `diff` is a string, it is used as a regular expression, to match with the names of the rows (or columns). Be precise enough to match only one column or row, otherwise you get a warning message.
- "no": not use diffs.

**comp**  The comparison level: by subtable/groups, or for the whole table. Vectorised over `row_vars`.
- "tab": by default, contributions to variance, row differences from totals/first cells, and row confidence intervals for these differences, are calculated for each `tab_vars` group.
- "all": compare cells to the general total line (provided there is a total table with a total row), or with the reference line of the total table when `diff = "first"`, an integer or a regular expression.

**chi2**  Set to `TRUE` to calculate Chi2 summaries with `tab_chi2`. Useful to print metadata, and to color cells based on their contribution to variance (color = "contrib"). Vectorised over `row_vars`.

**ci**  The type of confidence intervals to calculate, passed to `tab_ci`. Vectorised over `row_vars`.
- "cell": absolute confidence intervals of cells percentages.
tab_many

• "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when diff = "first").
• "auto": ci = "diff" for means and row/col percentages, ci = "cell" for frequencies ("all", "all_tabs").

By default, for percentages, with ci = "cell" Wilson’s method is used, and with ci = "diff" Wald’s method along Agresti and Caffo’s adjustment. Means use classic method. This can be changed in `tab_ci`.

conf_level
The confidence level, as a single numeric between 0 and 1. Default to 0.95 (95%).

method_cell
Character string specifying which method to use with percentages for ci = "cell". This can be one out of: "wald", "wilson", "wilsoncc", "agresti-coull", "jeffreys", "modified wilson", "modified jeffreys", "clopper-pearson", "arcsine", "logit", "witting", "pratt", "midp", "lik" and "blaker". Defaults to "wilson". See `BinomCI`.

method_diff
Character string specifying which method to use with percentages for ci = "diff". This can be one out of: "wald", "waldec", "ac", "score", "scorecc", "inn", "mee", "blj", "ha", "hal", "jp". Defaults to "ac", Wald interval with the adjustment according to Agresti, Caffo for difference in proportions and independent samples. See `BinomDiffCI`.

color
The type of colors to print, as a single string. Vectorised over row_vars.
• "no": by default, no colors are printed.
• "diff": color percentages and means based on cells differences from totals (or from first cells when diff = "first").
• "diff.ci": color pct and means based on cells differences from totals or first cells, removing coloring when the confidence interval of this difference is higher than the difference itself.
• "after.ci": idem, but cut off the confidence interval from the difference first.
• "contrib": color cells based on their contribution to variance (except mean columns, from numeric variables).
• "auto": frequencies (pct = "all", pct = "all_tabs") and counts are colored with "contrib". When ci = "diff", row and col percentages are colored with "after.ci"; otherwise they are colored with "diff".

subtext
A character vector to print rows of legend under the table.

clean_names
Set to TRUE to clean levels names, by removing prefix numbers like "1-", and text in parenthesis. All data formatting arguments are passed to `tab_prepare`.

rare_to_other
When set to TRUE, levels with less count than n.min will be merged into an "Other" level.

n.min
The count under which a level is aggregated in the "Other" level.

other_level
The name of the "Other" level, as a single string.

filter
A `dplyr::filter` to apply to the data frame first, as a single string (which will be converted to code, i.e. to a call). Useful when printing multiples tabs with `tibble::tribble`, to use different filters for similar tables or simply make the field of observation more visible into the code.

tabs
A tibble of class tab, made with `tab`, `tab_many` or `tab_plain`. 
In `tab_get_vars`, a character vector containing the wanted vars names: "row_var", "col_vars" or "tab_vars".

A object to test with `is_tab`.

Default to "positive", which just print breaks for positive spreads. Set to all to get breaks for negative spreads as well.

For `set_color_style` and `get_color_style`, is your console or html table background "light" or "dark"? Default to RStudio theme.

Should specific 24bits colors palettes be used for html tables? With light themes only. Default to `getOption("tabxplor.color_html_24_bit")`.

Possibility to provide a custom color styles, as a character vector of 10 html color codes (the five first for over-represented numbers, the five last for under-represented ones). The result is saved to options("tabxplor.color_style"). To discard, relaunch the function with `custom_palette = NULL`.

By default, `get_color_style` returns a list of `crayon` coloring functions. Set to "color_code" to return html color codes.

If they are to be changed, the breaks used for percentages. Default to `c(0.05, 0.1, 0.2, 0.3)`: first color used when the pct of a cell is +5% superior to the pct of the related total; second color used when it is +10% superior; third +20% superior; fourth +30% superior. The opposite for cells inferior to the total. With `color = "after_ci"`, the first break is subtracted from all breaks (default becomes `c(0, 0.05, 0.15, 0.25): +0%, +5%, +15%, +25%).

If they are to be changed, the breaks used for means. Default to `c(1.15, 1.5, 2, 4)`: first color used when the mean of a cell is superior to 1.15 times the mean of the related total row; second color used when it is superior to 1.5 times; etc. The opposite for cells inferior to the total. With `color = "after_ci"`, the first break is divided from all breaks (default becomes `c(1, 1.3, 1.7, 3.5)`).

If they are to be changed, the breaks used for contributions to variance. Default to `c(1, 2, 5, 10)`: first color used when the contribution of a cell is superior to the mean contribution; second color used when it is superior to 2 times the mean contribution; etc. The global color (for example green or red/orange) is given by the sign of the spread.

When missing, return all color breaks. Specify to return a given color break, among "pct", "mean", "contrib", "pct_ci" and "mean_ci".

A tibble of class `tab`, possibly with colored reading helpers. When there are two `row_vars` or more, a list of tibble of class `tab`. All non-text columns are of class `fmt`, storing all the data necessary to print formats and colors. Columns with `row_var` and `tab_vars` are of class `factor`: every added factor will be considered as a `tab_vars` and used for grouping. To add text columns without using them in calculations, be sure they are of class character.

A list with the variables names.

A single logical.

Set global options "tabxplor.color_style_type" and "tabxplor.color_style_theme", used when printing `tab` objects.
A vector of crayon color functions, or a vector of color html codes.

Set the global option "tabxplor.color_breaks" as a list different double vectors, and also returns it invisibly.

The color breaks as a double vector, or list of double vectors.

Functions

- `tab_get_vars()`: Get the variables names of a `tabxplor` tab
- `is_tab()`: a test function for class `tabxplor_tab`
- `set_color_style()`: define the color style used to print `tab`.
- `get_color_style()`: get color styles as `crayon` functions or html codes.
- `set_color_breaks()`: set the breaks used to print colors
- `get_color_breaks()`: get the breaks currently used to print colors

Examples

# Make a summary table with many col_vars, showing only one specific level:

```r
library(dplyr)
first_lvs <- c("Married", ",25000 or more", "Strong republican", "Protestant")
data <- forcats::gss_cat %>% mutate(across(
  where(is.factor),
  ~ forcats::fct_relevel(., first_lvs[first_lvs %in% levels(.)])
))
tab_many(data, race, c(marital, rincome, partyid, relig, age, tvhours),
  levels = "first", pct = "row", chi2 = TRUE, color = "auto")
```

# Can be used with map and tribble to program several tables with different parameters all at once, in a readable way:

```r
library(purrr)
library(tibble)

pmap(
  tribble(
    ~row_var, ~col_vars, ~pct, ~filter, ~subtext,
  ),
  .f = tab_many,
  data = forcats::gss_cat, color = "auto", chi2 = TRUE)

set_color_style(type = "bg")
set_color_breaks(
  pct_breaks = c(0.05, 0.15, 0.3),
  mean_breaks = c(1.15, 2, 4),
  contrib_breaks = c(1, 2, 5)
)
# tab_num

## Means table

### Description

Cross categorical variables with numeric variables, and get a table of means and standard deviations.

### Usage

```r
tab_num(
  data, 
  row_var, 
  col_vars, 
  tab_vars, 
  wt, 
  diff = "tot", 
  ci = NULL, 
  conf_level = 0.95, 
  comp = c("tab", "all"), 
  color = c("auto", "diff", "diff_ci", "after_ci"), 
  digits = 0, 
  na = c("keep", "drop", "drop_fct", "drop_num"), 
  totaltab = "line", 
  totaltab_name = "Ensemble", 
  tot = NULL, 
  total_names = "Total", 
  subtext = "", 
  num = FALSE, 
  df = FALSE 
)
```

### Arguments

- **data**: A data frame.
- **row_var**: The row variable, which will be printed with one level per line. If numeric, it will be used as a factor.
- **col_vars**: The numeric variables, which will appear in columns: means and standard deviation are calculated for each levels of `row_var` and `tab_vars`.
- **tab_vars**: `<tidy-select>` Tab variables: a subtable is made for each combination of levels of the selected variables. Leave empty to make a simple cross-table. All tab variables are converted to factor.
- **wt**: A weight variable, of class numeric. Leave empty for unweighted results.
- **diff**: The reference cell to calculate differences (used to print colors):
  - "tot": by default, cells differences from total rows are calculated with `pct = "row"`, and cells differences from total columns with `pct = "col"`. 

• "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).
• "no": not use diffs to gain calculation time.

ci The type of confidence intervals to calculate, passed to tab_ci (automatically added if needed for color).
• "cell": absolute confidence intervals of cells percentages.
• "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when diff = "first").
• "auto": ci = "diff" for means and row/col percentages, ci = "cell" for frequencies ("all", "all_tabs").

By default, for percentages, with ci = "cell" Wilson’s method is used, and with ci = "diff" Wald’s method along Agresti and Caffo’s adjustment. Means use classic method. This can be changed in tab_ci.

conf_level The confidence level for the confidence intervals, as a single numeric between 0 and 1. Default to 0.95 (95%).

comp Comparison level. When tab_vars are present, should the contributions to variance be calculated for each subtable/group (by default, comp = "tab")? Should they be calculated for the whole table (comp = "all")? comp must be set once and for all the first time you use tab_plain, tab_num or tab_chi2 with rows, or tab_ci.

color TRUE print the color percentages and means based on cells differences from totals or reference cell, as provided by diff. Default to FALSE, no colors.

digits The number of digits to print, as a single integer.

na The policy to adopt for missing values in row and tab variables (factors), as a single string.
• "keep": by default, NA’s of row and tab variables are printed as an explicit "NA" level.
• "drop": remove NA’s in row and tab variables.

NA’s in numeric variables are always removed when calculating means. For that reason the n field of each resulting fmt column, used to calculate confidence intervals, only takes into account the complete observations (without NA). To drop all rows with NA in any numeric variable first, use tab_prepare or tab_many with the na_drop_all argument.

tot The totals:
• c("col", "row") or "both": by default, both total rows and total columns.
• "row": only total rows.

totaltab The total table, if there are subtables/groups (i.e. when tab_vars is provided):
• "line": by default, add a general total line (necessary for calculations with comp = "all")
• "table": add a complete total table (i.e. row_var by col_vars without tab_vars).
• "no": not to draw any total table.

totaltab_name The name of the total table, as a single string.
• "col": only total column.
• "no": remove all totals (after calculations if needed).

total_names The names of the totals, as a character vector of length one or two. Use syntax of type c("Total row", "Total column") to set different names for rows and cols.

subtext A character vector to print rows of legend under the table.

num Set to TRUE to obtain a table with normal numeric vectors (not fmt).

df Set to TRUE to obtain a plain data.frame (not a tibble), with normal numeric vectors (not fmt). Useful, for example, to pass the table to correspondence analysis with FactoMineR.

Value
A tibble of class tabxplor_tab. If ... (tab_vars) are provided, a tab of class tabxplor_grouped_tab.
All non-text columns are fmt vectors of class tabxplor_fmt, storing all the data necessary to print formats and colors. Columns with row_var and tab_vars are of class factor: every added factor will be considered as a tab_vars and used for grouping. To add text columns without using them in calculations, be sure they are of class character.

Examples

data <- dplyr::storms %>% tab_prepare(category, wind, na_drop_all = wind)
tab_num(data, category, wind, tot = "row", color = "after_ci")

---

**tab_pct**

*Add percentages and diffs to a tab*

**Description**

Add percentages and diffs to a tab

**Usage**

```
tab_pct(
  tabs,
  pct = "row",
  digits = NULL,
  diff = c("tot", "first", "no"),
  comp = NULL,
  color = FALSE,
  just_diff = FALSE
)
```
Arguments

- **tabs**: A tibble of class tab made with `tab_plain` or `tab_many`.
- **pct**: The type of percentages to calculate. "row" draw row percentages. Set to "col" for column percentages. Set to "all" for frequencies (based on each subtable/group if `tab_vars` is provided). Set to "all_tabs" to calculate frequencies based on the whole (set of) table(s).
- **digits**: The number of digits to print for percentages. As a single integer, or an integer vector the same length than `col_vars`.
- **diff**: By default, with `pct = "row"`, differences from total rows are calculated, and with `pct = "col"` differences from total columns. Set to `diff = "first"` to calculate differences with the first cell of the row/col (useful to color temporal developments). When not using diffs for colors, set to `diff = "no"` to gain calculation time. Diffs are also calculated for mean columns (made from numeric variables).
- **comp**: Comparison level. When `tab_vars` are present, should the row differences be calculated for each subtable/group (by default `comp = "tab"`: comparison of each cell to the relative total row)? Should be calculated for the whole table (`comp = "all"`: comparison of each cell to the total row of the total table)? When `comp = "all"` and `diff = "first"`, cells are compared to the first cell of the total table instead. This parameter doesn’t affect column percentages. `comp` must be set once and for all the first time you use `tab_chisq`, `tab_pct` with rows, or `tab_ci`.
- **color**: Set to `TRUE` to color the resulting tab based on differences (from totals or from the first cell).
- **just_diff**: If percentages are already calculated and you just want to recalculate differences.

Value

A tibble of class tab, with percentages displayed, possibly colored based on differences from totals or first cell.

---

**tab_plain**

*Plain single cross-table*

---

**Description**

Plain single cross-table

**Usage**

```r
tab_plain(
  data,
  row_var,
  col_var,
  tab_vars,
```
Arguments

data A data frame.
row_var, col_var The row variable, which will be printed with one level per line, and the column variable, which will be printed with one level per column. Numeric variables will be used as factors. To calculate means, use `tab_num`.

`tab_vars` <tidy-select> Tab variables : a subtable is made for each combination of levels of the selected variables. Leave empty to make a simple cross-table. All tab variables are converted to factor.

wt A weight variable, of class numeric. Leave empty for unweighted results.

tabplain A data frame.

pct The type of percentages to calculate:
  - "row": row percentages.
  - "col": column percentages.
  - "all": frequencies for each subtable/group, if there is `tab_vars`.
  - "all-tabs": frequencies for the whole (set of) table(s).

diff The reference cell to calculate differences (used to print colors):
  - "tot": by default, cells differences from total rows are calculated with `pct = "row"`, and cells differences from total columns with `pct = "col"`.
  - "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).
  - "no": not use diffs to gain calculation time.

comp Comparison level. When `tab_vars` are present, should the contributions to variance be calculated for each subtable/group (by default, `comp = "tab"`)? Should they be calculated for the whole table (`comp = "all"`)? `comp` must be set once and for all the first time you use `tabPlain`, `tab_num` or `tab_chi2` with rows, or `tab_ci`.

color TRUE print the color percentages and means based on cells differences from totals or reference cell, as provided by `diff`. Default to FALSE, no colors.
**tab_plain**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>digits</td>
<td>The number of digits to print, as a single integer.</td>
</tr>
<tr>
<td>na</td>
<td>The policy to adopt with missing values, as a single string.</td>
</tr>
<tr>
<td>subtext</td>
<td>A character vector to print rows of legend under the table.</td>
</tr>
<tr>
<td>totaltab</td>
<td>The total table, if there are subtables/groups (i.e. when tab_vars is provided):</td>
</tr>
<tr>
<td>totaltab_name</td>
<td>The name of the total table, as a single string.</td>
</tr>
<tr>
<td>tot</td>
<td>The totals:</td>
</tr>
<tr>
<td>total_names</td>
<td>The names of the totals, as a character vector of length one or two. Use syntax of type c(&quot;Total row&quot;, &quot;Total column&quot;) to set different names for rows and cols.</td>
</tr>
<tr>
<td>num</td>
<td>Set to TRUE to obtain a table with normal numeric vectors (not fmt).</td>
</tr>
<tr>
<td>df</td>
<td>Set to TRUE to obtain a plain data.frame (not a tibble), with normal numeric vectors (not fmt). Useful, for example, to pass the table to correspondence analysis with FactoMineR.</td>
</tr>
</tbody>
</table>

**Value**

A tibble of class tabxplor_tab. If ... (tab_vars) are provided, a tab of class tabxplor_grouped_tab. All non-text columns are fmt vectors of class tabxplor_fmt, storing all the data necessary to print formats and colors. Columns with row_var and tab_vars are of class factor: every added factor will be considered as a tab_vars and used for grouping. To add text columns without using them in calculations, be sure they are of class character.

**Examples**

```r
# A typical workflow with tabxplor step-by-step functions :

data <- dplyr::starwars %>% tab_prepare(sex, hair_color)

data %>%
  tab_plain(sex, hair_color, tot = c("row", "col"), pct = "row") %>%
  tab_chi2() %>%
  tab_ci(color = "after_ci")
```
tab_prepare

Prepare data for tab_plain.

Description

Prepare data for tab_plain.

Usage

```r
tab_prepare(
  data,
  ..., 
  na_drop_all, 
  cleannames = NULL, 
  rare_to_other = FALSE, 
  n_min = 30, 
  other_level = "Others"
)
```

Arguments

- **data**
  A dataframe.
- **...**
  Variables then to be passed in tab_plain.
- **na_drop_all**
  <tidy-select> Removes all observation with a NA in any of the chosen variables.
- **cleannames**
  Set to TRUE to clean levels names, by removing prefix numbers like "1-", and text in parentheses.
- **rare_to_other**
  When set to TRUE, levels with less count than n_min will be merged into an "Other" level.
- **n_min**
  The count under which a level is aggregated in the "Other" level.
- **other_level**
  The name of the "Other" level, as a character vector of length one.

Value

A modified dataframe.

Examples

```r
data <- dplyr::starwars %>%
tab_prepare(sex, hair_color, gender, rare_to_other = TRUE,
  n_min = 5, na_drop_all = sex)
data
```
tab_spread

Spread a tab, passing a tab variable to column

Description

Spread a tab, passing a tab variable to column

Usage

```r
tab_spread(
  tabs,
  spread_vars,
  names_prefix = NULL,
  names_sort = FALSE,
  totname = "Total"
)
```

Arguments

- `tabs`: A tibble of class tab, made with `tab`, `tab_many` or `tab_plain`.
- `spread_vars`: `<tidy-select>` The tab variables to pass to column, with a syntax of type `c(var1, var2, ...)`.
- `names_prefix`: String added to the start of every variable name.
- `names_sort`: If no `names_prefix` is given, new names takes the form `spread_var_col_var_level`. Should then the column names be sorted? If `FALSE`, the default, column names are ordered by first appearance.
- `totname`: The new name of the total rows, as a single string.

Value

A tibble of class tab, with less rows and more columns.

Examples

```r
data <- forcats::gss_cat %>% dplyr::filter(year %in% c(2000, 2014))
tabs <-
  tab(data, relig, marital, c(year, race), pct = "row", totaltab = "no",
       color = "diff", tot = "row", rare_to_other = TRUE)

tabs %>%
  dplyr::select(year, race, relig, Married) %>%
  tab_spread(race)
```
### Description

Add totals to a tab

### Usage

```r
tab_tot(
  tabs,
  tot = c("row", "col"),
  name = "Total",
  totcol = "last",
  data = NULL
)
```

### Arguments

- **tabs**: A tibble of class tab, made with `tab_plain` or `tab_many`.
- **tot** *(c("col", "row") and "both" print total rows and total columns. Set to "row" or "col" to print only one type. Set to "no" to remove all totals.)*
- **name** *(The names of the totals, as a character vector of length one or two. Use c("Total_row", "Total_column") to set different names for rows and cols.)*
- **totcol** *("last" only prints a total column for the last factor column variable. Set to "each" to print a total column for each column variable.)*
- **data** *(The original database used to calculate the tab: it is only useful for mean columns (of numeric variables), in order to calculate the variances of total rows, necessary to calculate confidence intervals with `tab_ci`.)*

### Value

A tibble of class tab. Total rows can then be detected using `is_totrow`, and total columns using `is_totcol`.

### Examples

```r
data <- dplyr::starwars %>% tab_prepare(sex, hair_color)
data %>%
  tab_plain(sex, hair_color) %>%
tab_tot("col", totcol = "each")
```
Add total table to a tab

Description

Add total table to a tab

Usage

```r
tab_totaltab(
  tabs,
  totaltab = c("table", "line", "no"),
  name = "Ensemble",
  data = NULL
)
```

Arguments

- **tabs**: A tibble of class tab, made with `tab_plain` or `tab_many`.
- **totaltab**: If there are subtables, corresponding to the levels of `tab_vars`, `totaltab = "table"` add a complete total table. `totaltab = "line"` add a total table of only one row with the general total. `totaltab = "no"` remove any existing total table.
- **name**: The name of the total table, as a single string.
- **data**: The original database used to calculate the `tab` : it is only useful for mean columns (of numeric variables), in order to calculate the variances necessary to calculate confidence intervals with `tab_ci`.

Value

A tibble of class tab. Rows belonging to the total table can then be detected using `is_tottab`.

Examples

```r
data <- dplyr::starwars %>%
  tab_prepare(sex, hair_color, gender, rare_to_other = TRUE,
              n_min = 5, na_drop_all = sex)

data %>%
  tab_plain(sex, hair_color, gender) %>%
  tab_totaltab("line")
```
Description

To modify the colors used into the Excel table, you can change the global options with `set_color_style` and `set_color_breaks`.

Usage

```r
tab_xl(
  tabs,
  path = NULL,
  replace = FALSE,
  open = rlang::is_interactive(),
  colnames_rotation = 0,
  remove_tab_vars = TRUE,
  colwidth = "auto",
  print_ci = FALSE,
  print_color_legend = TRUE,
  sheets = "tabs",
  n_min = 0,
  titles,
  hide_near_zero = "auto",
  color_type = "text"
)
```

Arguments

- **tabs**: A table made with `tab`, `tab_many` or `tab_plain`, or a list of such tables.
- **path**, **replace**, **open**
  - The name, and possibly the path, of the Excel file to create (possibly without the .xlsx extension). Default path to temporary directory. Set global option "tabxplor.export_dir" with `link[base:options]{options}` to change default directory. By default replace is TRUE when path is provided, FALSE when path is not provided. Use replace = TRUE to overwrite existing files. Use open = FALSE if you don’t want to automatically open the tables in Excel (or another software associated with .xlsx files).
- **colnames_rotation**: Rotate the names of columns to an angle (in degrees).
- **remove_tab_vars**: By default, tab_vars columns are removed to gain space. Set to FALSE to keep them.
- **colwidth**: The standard width for numeric columns, as a number. Default to "auto".
- **print_ci**: Set to TRUE to print confidence intervals in another table, at the left of the base table.
print_color_legend
Should the color legends be printed with the subtexts?

sheets
The Excel sheets options:
- "tabs": a new sheet is created for each table
- "unique": all tables are on the same sheet
- "auto": subsequent tables with the same columns are printed on the same sheets

n_min
The total count under which a column or row is turned pale grey because there is not enough observation for it to be significant. Default to 0 (not used).

titles
The titles of the different tables, as a character vector. When missing titles are given based on the names of the variables.

hide_near_zero
By default all cells displayed as 0 (even rounded) turn pale grey, to make the distribution of empty cells (and other cells) more visible. Provide a number to turn grey every cell below it. Set to Inf not to use this feature.

color_type
By default, the text is colored. Set to "bg" to color the background instead.

Value
The table(s) with formatting and colors in an Excel file, as a side effect. Invisibly returns tabs.

Examples

```r
tab_xlConfidential <-
forcats::gss_cat %>%
  tab(marital, race, pct = "row", color = "diff") %>%
  tab_xl()
```

Description
Excel output for tabxplor tables with confidentiality rules. Don’t forget to provide subtext = c("Source : description of the source of the data") in tab or tab_many, otherwise it is not possible to assess, for your reader, which confidentiality rules applies. For the same reason, you must supply a description of all variables in var_labels.

Usage

```r
tab_xl_confidential(
  tabs,  
  path = NULL,  
  replace = NULL,  
  open = rlang::is_interactive(),
)```
n_min = 5,
pct_max = 0.95,
recalculate_totcols = NULL,
var_labels = character(),
colnames_rotation = 0,
colwidth = 10,
sheets = "unique",
print_color_legend = TRUE,
titles,
hide_near_zero = "auto",
color_type = "text"
)

Arguments

tabs A table made with tab, tab_many or tab_plain, or a list of such tables.

path, replace, open The name, and possibly the path, of the Excel file to create (possibly without the .xlsx extension). Default path to temporary directory. Set global option "tabxplor.export_dir" with link[base:options](options) to change default directory. By default replace is TRUE when path is provided, FALSE when path is not provided. Use replace = TRUE to overwrite existing files. Use open = FALSE if you don't want to automatically open the tables in Excel (or another software associated with .xlsx files).

n_min The total count under which a column or row doesn't respect statistical confidentiality. Default to 5.

pct_max The row or column percentage above which, knowing the column category, it becomes possible to guess the row category, or the other way round. Default to 0.95 (95%).

recalculate_totcols By default, total columns are recalculated from counts if there are many col_vars but only one total column. Provide a logical vector the length of the number of tables, or a single logical, to choose the wanted behavior. The fastest way to do it is to use tab_many() with totcol = "each" before.

var_labels The description of all the variables, necessary to assess that the tables don’t break confidentiality rules, as a character vector of the type c('variable1' = 'description of the variable', 'variable2' = ...)

colnames_rotation Rotate the names of columns to an angle (in degrees).

colwidth The standard width for numeric columns, as a number. Default to 10.

sheets The Excel sheets options :
  • "unique": all tables are on the same sheet
  • "tabs": a new sheet is created for each table
  • "auto": subsequent tables with the same columns are printed on the same sheets

print_color_legned Should the color legends be printed with the subtexts?
The titles of the different tables, as a character vector. When missing titles are given based on the names of the variables.

hide_near_zero
By default all cells displayed as 0 (even rounded) turn pale grey, to make the distribution of empty cells (and other cells) more visible. Provide a number to turn grey every cell below it. Set to Inf not to use this feature.

color_type
By default, the text is colored. Set to "bg" to color the background instead.

Value
The table(s) with formatting and colors in an Excel file, as a side effect. Invisibly returns tabs.

Examples

```r
forcats::gss_cat |>
  tab(race, marital, year, pct = "row", color = "diff",
  subtext = c("Source: National Opinion Research Center, General Social Survey.'\n))) |>
  tab_xl_confidential(titles = "Marital status by race",
  var_labels = c("marital" = "marital status", "race" = "race",
  "year" = "year of survey"))
```

Description
Table body for class tab

Usage

```r
## S3 method for class 'tabxplor_tab'
.tbl_format_body(x, setup, ...)
```

Arguments

- `x` An object of class tabxplor_tab
- `setup` A setup object from the table
- `...` Other parameters.

Value
A character vector.
## Description

Table footer for class `tab`.

## Usage

```r
## S3 method for class 'tabxplor_tab'
tbl_format_footer(x, setup, ...)
```

### Arguments

- `x`: An object of class `tabxplor_tab`
- `setup`: A setup object from the table
- `...`: Other parameters.

### Value

A character vector.

## Description

Table headers for class `grouped_tab`.

## Usage

```r
## S3 method for class 'tabxplor_grouped_tab'
tbl_sum(x, ...)
```

### Arguments

- `x`: An object of class `tabxplor_grouped_tab`
- `...`: Other parameters.

### Value

A table header
### Description

Table headers for class `tabxplor_tab`  

#### Usage

```r
## S3 method for class 'tabxplor_tab'
.tbl_sum(x, ...)
```

#### Arguments

- `x`: An object of class `tabxplor_tab`
- `...`: Other parameters.

#### Value

A table header

---

### Description

Ungroup method for class `tabxplor_grouped_tab`

#### Usage

```r
## S3 method for class 'tabxplor_grouped_tab'
.ungroup(x, ...)
```

#### Arguments

- `x`: A tibble of class `tabxplor_grouped_tab`
- `...`: Variables to remove from the grouping.

#### Value

An object of class `tabxplor_tab` or `tabxplor_grouped_tab`. 
vec_arith.tabxplor_fmt

Vec_arith method for fmt

Description

Vec_arith method for fmt

Usage

## S3 method for class 'tabxplor_fmt'
vec_arith(op, x, y, ...)

## Default S3 method:
vec_arith.tabxplor_fmt(op, x, y, ...)

## S3 method for class 'tabxplor_fmt'
vec_arith.tabxplor_fmt(op, x, y, ...)

## S3 method for class 'numeric'
vec_arith.tabxplor_fmt(op, x, y, ...)

## S3 method for class 'MISSING'
vec_arith.tabxplor_fmt(op, x, y, ...)

Arguments

op Operation to do.
x fmt object.
y Second object.
... Other parameter.

Value

A fmt vector
A fmt vector
A fmt vector
A fmt vector
A fmt vector
A fmt vector
A fmt vector
A fmt vector
Methods (by class)

• vec_arith.tabxplor_fmt(default): default vec_arith method for fmt
• vec_arith.tabxplor_fmt(tabxplorFmt): vec_arith method for fmt + fmt
• vec_arith.tabxplor_fmt(numeric): vec_arith method for fmt + numeric
• vec_arith.tabxplor_fmt(MISSING): vec_arith method for -fmt

Functions

• vec_arith.numeric(tabxplorFmt): vec_arith method for numeric + fmt

Description

Convert fmt into character

Usage

## S3 method for class 'tabxplor_FMT'
vec_cast.character(x, to, ...)

Arguments

x  A fmt vector

to A character vector

... Other parameter

Value

A character vector
vec_cast.double.tabxplor_fmt

Convert fmt into double

Description
Convert fmt into double

Usage
## S3 method for class 'tabxplor_fmt'
vec_cast.double(x, to, ...)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A fmt vector</td>
</tr>
<tr>
<td>to</td>
<td>A double vector</td>
</tr>
<tr>
<td>...</td>
<td>Other parameter.</td>
</tr>
</tbody>
</table>

Value
A double vector

vec_cast.integer.tabxplor_fmt

Convert fmt into integer

Description
Convert fmt into integer

Usage
## S3 method for class 'tabxplor_fmt'
vec_cast.integer(x, to, ...)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A integer vector</td>
</tr>
<tr>
<td>to</td>
<td>A fmt vector</td>
</tr>
<tr>
<td>...</td>
<td>Other parameter.</td>
</tr>
</tbody>
</table>

Value
An integer vector
vec_cast.tabxplor_fmt.double

Convert double into fmt

Description
Convert double into fmt

Usage
## S3 method for class 'tabxplor_fmt.double'
vec_cast(x, to, ...)

Arguments
- **x**  A double vector
- **to** A fmt vector
- **...** Other parameter.

Value
A fmt vector

vec_cast.tabxplor_fmt.integer

Convert integer into fmt

Description
Convert integer into fmt

Usage
## S3 method for class 'tabxplor_fmt.integer'
vec_cast(x, to, ...)

Arguments
- **x**  A integer vector
- **to** A fmt vector
- **...** Other parameter.

Value
A fmt vector
vec_cast.tabxplor_fmt
Convert fmt into fmt

Description
Convert fmt into fmt

Usage
## S3 method for class 'tabxplor_fmt.tabxplor_fmt'
vec_cast(x, to, ...)

Arguments
x  A fmt vector
  to  A fmt vector
...
Other parameter.

Value
A fmt vector

vec_math.tabxplor_fmt  Vec_math method for class fmt

Description
Vec_math method for class fmt

Usage
## S3 method for class 'tabxplor_fmt'
vec_math(.fn, .x, ...)

Arguments
.fn  A function
  .x  A fmt object
...
Other parameter

Value
A fmt vector
vec_proxy_compare.tabxplor_fmt

Compare with fmt vector

Description

Compare with fmt vector

Usage

## S3 method for class 'tabxplor_fmt'
vec_proxy_compare(x, ...)

Arguments

x  A fmt vector
...

Value

A double vector

vec_proxy_equal.tabxplor_fmt

Test equality with fmt vector

Description

Test equality with fmt vector

Usage

## S3 method for class 'tabxplor_fmt'
vec_proxy_equal(x, ...)

Arguments

x  A fmt vector
...

Value

A double vector
vec_ptype2(double.tabxplorFmt)

Find common ptype between double and fmt

Description
Find common ptype between double and fmt

Usage
## S3 method for class 'double.tabxplorFmt'
vec_ptype2(x, y, ...)

Arguments
x A double vector
y A fmt vector
... Other parameter.

Value
A fmt vector

vec_ptype2(integer.tabxplorFmt)

Find common ptype between integer and fmt

Description
Find common ptype between integer and fmt

Usage
## S3 method for class 'integer.tabxplorFmt'
vec_ptype2(x, y, ...)

Arguments
x An integer vector
y A fmt vector
... Other parameter.

Value
A fmt vector
vec_ptype2.tabxplor_fmt.double

Find common ptype between fmt and double

Description
Find common ptype between fmt and double

Usage
## S3 method for class 'tabxplor_fmt.double'
vec_ptype2(x, y, ...)

Arguments
- x: A fmt vector
- y: A double vector
- ...: Other parameter.

Value
A fmt vector

vec_ptype2.tabxplor_fmt.integer

Find common ptype between fmt and integer

Description
Find common ptype between fmt and integer

Usage
## S3 method for class 'tabxplor_fmt.integer'
vec_ptype2(x, y, ...)

Arguments
- x: A fmt vector
- y: An integer vector
- ...: Other parameter.

Value
A fmt vector
vec_ptype2.tabxplor_fmt.tabxplor_fmt

Find common ptype between fmt and fmt

Description
Find common ptype between fmt and fmt

Usage
```r
## S3 method for class 'tabxplor_fmt.tabxplor_fmt'
vec_ptype2(x, y, ...)
```

Arguments
- `x`: A fmt object.
- `y`: A fmt object.
- `...`: Other parameter.

Value
A fmt vector

vec_ptype_abbr.tabxplor_fmt

Abbreviated display name for class fmt in tibbles

Description
Abbreviated display name for class fmt in tibbles

Usage
```r
## S3 method for class 'tabxplor_fmt'
vec_ptype_abbr(x, ...)
```

Arguments
- `x`: A fmt object.
- `...`: Other parameter.

Value
A single string with abbreviated fmt type.
vec_ptype_full.tabxplor_fmt

*Printed type for class fmt*

**Description**

Printed type for class fmt

**Usage**

```r
## S3 method for class 'tabxplor_fmt'
vec_ptype_full(x, ...)
```

**Arguments**

- `x`: A fmt object.
- `...`: Other parameter.

**Value**

A single string with full fmt type.

---

`.tabxplor_grouped_tab

*subset method for class tabxplor_grouped_tab*

**Description**

subset method for class tabxplor_grouped_tab

**Usage**

```
"x[i] ; x[i, j, ... , drop = TRUE]"
```

**Arguments**

- `x`: A tabxplor_grouped_tab object.
- `i, j, ...`: Indices
- `drop`: For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement.

**Value**

An object of class tabxplor_grouped_tab.
set subset method for class `tabxplor_grouped_tab`

**Description**

set subset method for class `tabxplor_grouped_tab`

**Usage**

```
"x[i] <- value ; x[i, j, ...] <- value"
```

**Arguments**

- `x`: A `tabxplor_grouped_tab` object.
- `i, j, ...`: Indices.
- `value`: The new value.

**Value**

An object of class `tabxplor_grouped_tab`.

[[<-.`tabxplor_grouped_tab

set sub-subset method for class `tabxplor_grouped_tab`

**Description**

set sub-subset method for class `tabxplor_grouped_tab`

**Usage**

```
"x[[...]] <- value"
```

**Arguments**

- `x`: A `tabxplor_grouped_tab` object.
- `...`: Indices
- `value`: The new value.

**Value**

An object of class `tabxplor_grouped_tab`. 
Description

$ method for class tabxplorFmt

Usage

## S3 method for class `tabxplorFmt`

```r
x$name
```

Arguments

- **x**: A `tabxplorFmt` object.
- **name**: The name of the field to extract.

Value

The relevant field of the `tabxplorFmt`. 
Index

['.tabxplor_grouped_tab, 77
['<-.*tabxplor_grouped_tab, 78
[['<-.*tabxplor_grouped_tab, 78
$.tabxplor_fmt, 79

arrange, 37, 45
as_refcol(fmt), 6
as_refrow(fmt), 6
as_totcol(fmt), 6
as_totrow(fmt), 6
as_tottab(fmt), 6
attr, 6

BinomCI, 44, 49
BinomDiffCI, 44, 49

cmplete_partial_totals, 4
dplyr::filter, 40, 49
dplyr::select, 47
dplyr_col_modify.tabxplor_grouped_tab, 4
dplyr_reconstruct.tabxplor_grouped_tab, 5
dplyr_row_slice.tabxplor_grouped_tab, 5

fct_recode_helper, 6
filter, 37, 45
fmt, 6, 9, 40, 50, 53, 54, 57
fmt_get_color_code, 13
format.tabxplor_fmt, 14

get_ci_type(fmt), 6
get_ci_type.data.frame, 14
get_ci_type.default, 15
get_ci_type.tabxplor_fmt, 15
get_col_var(fmt), 6
get_col_var.data.frame, 17
get_col_var.default, 18

get_col_var.tabxplor_fmt, 18

get_color(fmt), 6
get_color.data.frame, 16
get_color.default, 16
get_color.tabxplor_fmt, 17
get_color_breaks(tab_many), 45
get_color_style(tab_many), 45
get_comp_all, 10
get_comp_all(fmt), 6
get_diff_type(fmt), 6
get_diff_type.data.frame, 19
get_diff_type.default, 19
get_diff_type.tabxplor_fmt, 20
get_digits(fmt), 6
get_num(fmt), 6
get_type(fmt), 6
get_type.data.frame, 20
get_type.default, 21
get_type.tabxplor_fmt, 21
group_by.tabxplor_tab, 22

is_fmt(fmt), 6
is_refcol(fmt), 6
is_refcol.data.frame, 22
is_refcol.default, 23
is_refcol.tabxplor_fmt, 23
is_refrow(fmt), 6
is_refrow.data.frame, 24
is_refrow.default, 24
is_refrow.tabxplor_fmt, 25

is_tab, 50
is_tab.data.frame, 50
is_tab.tab_many, 45
is_totcol, 60
is_totcol.data.frame, 6
is_totcol.default, 6
is_totcol.tabxplor_fmt, 26
is_totrow, 60
is_totrow.data.frame, 27
is_totrow.default, 27
INDEX

is_totrow.tabxplor_fmt, 28
is_tottab, 61
is_tottab(fmt), 6
is_tottab.data.frame, 28
is_tottab.default, 29
is_tottab.tabxplor_fmt, 29

kableExtra::kable_styling, 45
mutate, 37, 45
mutate.tabxplor_fmt, 30

new_grouped_tab(new_tab), 30
new_tab, 30

pillar_shaft.tab.chi2_fmt, 32
pillar_shaft.tabxplor_fmt, 31
print.tabxplor_grouped_tab, 32
print.tabxplor_tab, 33

record, 6
relocate.tabxplor_grouped_tab, 34
rename.tabxplor_grouped_tab, 34
rename_with.tabxplor_grouped_tab, 35
rowwise.tabxplor_grouped_tab, 35
rowwise.tabxplor_tab, 36

select, 37, 45
select.tabxplor_grouped_tab, 36
set_ci_type(fmt), 6
set_col_var(fmt), 6
set_color(fmt), 6
set_color_breaks, 62
set_color_breaks(tab_many), 45
set_color_style, 62
set_color_style(tab_many), 45
set_comp_all(fmt), 6
set_diff_type(fmt), 6
set_digits(fmt), 6
set_num(fmt), 6
set_type(fmt), 6
summarise.tabxplor_grouped_tab, 37

tab, 6, 31, 37, 42, 43, 45, 49–51, 54, 59–62, 64
tab.chi2, 9, 31, 40, 42, 43, 48, 53, 55, 56
tab.ci, 9, 40, 42, 43, 48, 49, 53, 55, 56,
60, 61

tab.get-vars(tab_many), 45
tab_kable, 44