

# Package ‘tabxplor’

June 16, 2022

**Title** User-Friendly Tables with Color Helpers for Data Exploration

**Version** 1.1.0

**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** 7.1.2

**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** 2022-06-15 22:50:02 UTC

**R topics documented:**

dplyr_col_modify.tabxplor_grouped_tab . . . . .	4
dplyr_reconstruct.tabxplor_grouped_tab . . . . .	4
dplyr_row_slice.tabxplor_grouped_tab . . . . .	5
fct_recode_helper . . . . .	5
fmt . . . . .	6
fmt_get_color_code . . . . .	13
format.tabxplor_fmt . . . . .	13
get_ci_type.data.frame . . . . .	14
get_ci_type.default . . . . .	14
get_ci_type.tabxplor_fmt . . . . .	15
get_color.data.frame . . . . .	15
get_color.default . . . . .	16
get_color.tabxplor_fmt . . . . .	16
get_col_var.data.frame . . . . .	17
get_col_var.default . . . . .	17
get_col_var.tabxplor_fmt . . . . .	18
get_diff_type.data.frame . . . . .	18
get_diff_type.default . . . . .	19
get_diff_type.tabxplor_fmt . . . . .	19
get_type.data.frame . . . . .	20
get_type.default . . . . .	20
get_type.tabxplor_fmt . . . . .	21
group_by.tabxplor_tab . . . . .	21
is_refcol.data.frame . . . . .	22
is_refcol.default . . . . .	22
is_refcol.tabxplor_fmt . . . . .	23
is_refrow.data.frame . . . . .	23
is_refrow.default . . . . .	24
is_refrow.tabxplor_fmt . . . . .	24
is_totcol.data.frame . . . . .	25
is_totcol.default . . . . .	25
is_totcol.tabxplor_fmt . . . . .	26
is_totrow.data.frame . . . . .	26
is_totrow.default . . . . .	27
is_totrow.tabxplor_fmt . . . . .	27
is_tottab.data.frame . . . . .	28
is_tottab.default . . . . .	28
is_tottab.tabxplor_fmt . . . . .	29
mutate.tabxplor_fmt . . . . .	29
new_tab . . . . .	30
pillar_shaft.tabxplor_fmt . . . . .	31
pillar_shaft.tab_chi2_fmt . . . . .	31
print.tabxplor_grouped_tab . . . . .	32
print.tabxplor_tab . . . . .	33
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.tabxplor_grouped_tab . . . . .	36
summarise.tabxplor_grouped_tab . . . . .	37
tab . . . . .	37
tab_chi2 . . . . .	42
tab_ci . . . . .	43
tab_kable . . . . .	44
tab_many . . . . .	46
tab_num . . . . .	52
tab_pct . . . . .	55
tab_plain . . . . .	56
tab_prepare . . . . .	58
tab_spread . . . . .	59
tab_tot . . . . .	60
tab_totaltab . . . . .	61
tab_xl . . . . .	62
tbl_format_body.tabxplor_tab . . . . .	63
tbl_format_footer.tabxplor_tab . . . . .	64
tbl_sum.tabxplor_grouped_tab . . . . .	64
tbl_sum.tabxplor_tab . . . . .	65
ungroup.tabxplor_grouped_tab . . . . .	65
vec_arith.tabxplor_fmt . . . . .	66
vec_cast.character.tabxplor_fmt . . . . .	67
vec_cast.double.tabxplor_fmt . . . . .	67
vec_cast.integer.tabxplor_fmt . . . . .	68
vec_cast.tabxplor_fmt.double . . . . .	68
vec_cast.tabxplor_fmt.integer . . . . .	69
vec_cast.tabxplor_fmt.tabxplor_fmt . . . . .	69
vec_math.tabxplor_fmt . . . . .	70
vec_proxy_compare.tabxplor_fmt . . . . .	70
vec_proxy_equal.tabxplor_fmt . . . . .	71
vec_ptype2.double.tabxplor_fmt . . . . .	71
vec_ptype2.integer.tabxplor_fmt . . . . .	72
vec_ptype2.tabxplor_fmt.double . . . . .	72
vec_ptype2.tabxplor_fmt.integer . . . . .	73
vec_ptype2.tabxplor_fmt.tabxplor_fmt . . . . .	73
vec_ptype_abbrev.tabxplor_fmt . . . . .	74
vec_ptype_full.tabxplor_fmt . . . . .	74
\$.tabxplor_fmt . . . . .	75

---

```
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

```
## 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.tabxplor_grouped_tab
```

```
dplyr_reconstruct method for class tabxplor_grouped_tab
```

---

### Description

dplyr\_reconstruct method for class tabxplor\_grouped\_tab

### Usage

```
## 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.tabxplor_grouped_tab
      dplyr_row_slice method for class tabxplor_grouped_tab
```

---

**Description**

dplyr\_row\_slice method for class tabxplor\_grouped\_tab

**Usage**

```
## 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	<a href="#">&lt;tidy-select&gt;</a> 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**

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 `tabxplore_fmt`, powers **tabxplore** and `tab` tibbles. As a [record](#), they stores 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 **tabxplore** formatted numbers. `taxplfmt` vectors can use all standard operations, like `+`, `-`, `sum()`, or `c()`, using `vctrs`.

**Usage**

```
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(FALSE, 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 <code>n()</code> . 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) : <ul style="list-style-type: none"> <li>• "n": counts</li> <li>• "mean": mean column (from numeric variables)</li> <li>• "row": row percentages</li> <li>• "col": column percentages</li> <li>• "all": frequencies by subtable/group (i.e. by <code>tab_vars</code>)</li> <li>• "all_tabs": frequencies for the whole table</li> </ul>
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 <code>fmt</code> , like means.
pct	The percentages, as a double vector the length of n. Calculate with <code>tab_pct</code> .
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 <code>tab_pct</code> .
ctr	The contributions of cells to (sub)tables variances, as a double vector the length of n. Used to print colors when <code>color = "contrib"</code> . The mean contribution of each (sub)table is written on total rows (then, colors don't print well without total rows). Calculate with <code>tab_chi2</code> .
var	The cells variances, as a double vector the length of n. Used with <code>type = "mean"</code> to calculate confidence intervals. Calculate with <code>tab_plain</code> .
ci	The confidence intervals, as a double vector the length of n. Used to print colors ("diff_ci", "after_ci"). Calculate with <code>tab_ci</code> .
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. <code>diff_type</code> )
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 <code>tab_pct</code> ): <ul style="list-style-type: none"> <li>• "" or "no": no differences have been calculated</li> <li>• "tot": the reference row (or column) is the total row (or column)</li> <li>• "first": the reference row (or column) is the first row (or column)</li> </ul>
ci_type	The type of confidence intervals of the vector (calculate with <code>tab_ci</code> ): <ul style="list-style-type: none"> <li>• "" or "no": no ci have been calculated</li> <li>• "cell": absolute confidence intervals of cells percentages.</li> <li>• "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when <code>diff_type = "first"</code>).</li> <li>• "auto": "diff" for means and row/col percentages, "cell" for frequencies ("all", "all_tabs").</li> </ul>
col_var	The name of the <code>col_var</code> 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 : <ul style="list-style-type: none"> <li>• "no": no colors are printed.</li> <li>• "diff": color percentages and means based on cells differences from totals (or from first cells when <code>diff = "first"</code>).</li> <li>• "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.</li> <li>• "after_ci": idem, but cut off the confidence interval from the difference first.</li> <li>• "contrib": color cells based on their contribution to variance (except mean columns, from numeric variables).</li> </ul>
x	The object to test, to get a field in, or to modify.
value	The value you want to inject in some <code>fmt</code> vector's <code>vctrs::field</code> or attribute using a given "set" function.
...	Used in methods to add arguments in the future.
replace_na	By default, <code>get_comp_all</code> takes NA in comparison level to be a FALSE (=comparison at subtables/groups level). Set to FALSE to avoid this behavior.

### Value

A vector of class `tabxplore_fmt`.

A logical vector.

A double vector.

A modified `fmt` vector.

A character vector with the vectors type.

A modified `fmt` vector.

A logical vector with the `fmt` vectors `totrow` field.

A modified fmt vector with totrow field changed.

A logical vector with the fmt vectors tottab field.

A modified fmt vector with tottab field changed.

A logical vector with the fmt vectors totcol attribute.

A modified fmt vector with totcol attribute changed.

A logical vector with the fmt vectors in\_refrow field

A modified fmt vector with in\_refrow field changed.

A modified fmt vector with comp attribute changed.

A logical vector with the fmt vectors type attributes

A modified fmt vector.

A logical vector with the fmt vectors ci\_type attributes

A modified fmt vector.

A logical vector with the fmt vectors col\_var attributes

A modified fmt vector.

A logical vector with the fmt vectors is\_refcol attributes

A modified fmt vector.

A logical vector with the fmt vectors color attributes

A modified 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) :
vctrs::vec_data(f)

# To get the numbers of digits :
vctrs::field(f, "digits")
f$digits

# To get the count :
vctrs::field(f, "n")
f$n

# To get the display :
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`) :
vctrs::`field`<-(f, "pct", c(1, 0, 0))
mutate(f, pct = c(1, 0, 0))

# See all the attributes of a fmt vector :
attributes(f)
```

```

# To modify the "type" attribute of a fmt vector :
set_type(f, "col")

# To modify the "color" attribute of a fmt vector :
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 :
is_totcol(tabs)
tabs %>% mutate(across(where(is_totcol), ~ "total column"))

# To identify the total rows, and work with them :
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 :
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(., #Mutate over each fmt vector's underlying data.frame.
      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, ...)
```

**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**

```
## 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**

```
## 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.

**Value**

A single character with the ci\_type attribute.

---

get\_color.data.frame                    *Get color*

---

**Description**

Get color

**Usage**

```
## 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**

```
## 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**

```
## S3 method for class 'tabxplor_fmt'
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_col_var.data.frame`*Get names of column variable of fmt columns*

---

**Description**

Get names of column variable of fmt columns

**Usage**

```
## S3 method for class 'data.frame'  
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 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
```

*Get names of column variable of fmt columns*

---

### 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**

```
## 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**

```
## 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        *Get types of fmt columns*

---

**Description**

Get types of fmt columns

**Usage**

```
## 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 *Get types of fmt columns*

---

### Description

Get types of fmt columns

### Usage

```
## 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

```
## 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**

```
## 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**

```
## 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**

```
## 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**

```
## 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**

```
## 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
```

*Test function for total columns*

---

### 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.  
 ...                  Used in methods to add arguments in the future.  
 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, ...)
```

**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\_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.

---

is\_tottab.default    *Test function to detect cells in total tables*

---

### 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**

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

**Arguments**

x                    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**

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

```
new_grouped_tab(
  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 <a href="#">tibble</a> data.frame. It is generally made with <a href="#">tab</a> , <a href="#">tab_many</a> or <a href="#">tab_plain</a> .
subtext	A character vector to print legend lines under the table.
chi2	A tibble storing information about pvalues and variances, to fill with <a href="#">tab_chi2</a> .
...	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**

```
## 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  
)
```

## 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.

## Value

A printed table.

---

relocate.tabxplor\_grouped\_tab  
*relocate method for class tabxplor\_grouped\_tab*

---

**Description**

relocate method for class tabxplor\_grouped\_tab

**Usage**

```
## 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  
*rename method for class tabxplor\_grouped\_tab*

---

**Description**

rename method for class tabxplor\_grouped\_tab

**Usage**

```
## 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.

---

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

**Value**

A tibble of class tabxplor\_grouped\_tab and rowwise\_df.

---

select.tabxplor\_grouped\_tab  
*select method for class 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_tab.
...	One or more unquoted expressions separated by commas. Variable names can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of variables.

**Value**

An object of class tabxplor\_grouped\_tab.

---

```
summarise.tabxplor_grouped_tab
  summarise method for class tabxplor_grouped_tab
```

---

**Description**

summarise method for class tabxplor\_grouped\_tab

**Usage**

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

**Arguments**

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

**Value**

An object of class `tabxplor_grouped_tab`.

---

<code>tab</code>	<i>Single cross-table, with color helpers</i>
------------------	---

---

**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	<a href="#">&lt;tidy-select&gt;</a> 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	<a href="#">&lt;tidy-select&gt;</a> 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 <code>sup_cols = c(sup_col1, sup_col2, ...)</code> . To keep all levels of other col_vars, or other types of percentages, use <a href="#">tab_many</a> instead.
na	The policy to adopt for missing values, as a single string : <ul style="list-style-type: none"> <li>• "keep": by default, NA's of row, col and tab variables are printed as an explicit "NA" level.</li> <li>• "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.</li> </ul>
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 : <ul style="list-style-type: none"> <li>• "row": row percentages.</li> </ul>

	<ul style="list-style-type: none"> <li>• "col": column percentages.</li> <li>• "all": frequencies for each subtable/group, if there is tab_vars.</li> <li>• "all_tabs": frequencies for the whole (set of) table(s).</li> </ul>
color	<p>The type of colors to print, as a single string :</p> <ul style="list-style-type: none"> <li>• "no": by default, no colors are printed.</li> <li>• "diff": color percentages and means based on cells differences from totals (or from first cells when diff = "first").</li> <li>• "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.</li> <li>• "after_ci": idem, but cut off the confidence interval from the difference first.</li> <li>• "contrib": color cells based on their contribution to variance (except mean columns, from numeric variables).</li> <li>• "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".</li> </ul>
diff	<p>The reference cell to calculate differences (used to print colors) :</p> <ul style="list-style-type: none"> <li>• "tot": by default, cells differences from total rows are calculated with pct = "row", and cells differences from total columns with pct = "col".</li> <li>• "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).</li> <li>• "no": not use diffs to gain calculation time.</li> </ul>
comp	<p>The comparison level : by subtables/groups, or for the whole table.</p> <ul style="list-style-type: none"> <li>• "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.</li> <li>• "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".</li> </ul>
totaltab	<p>The total table, if there are subtables/groups (i.e. when tab_vars is provided) :</p> <ul style="list-style-type: none"> <li>• "line": by default, add a general total line (necessary for calculations with comp = "all")</li> <li>• "table": add a complete total table (i.e. row_var by col_vars without tab_vars).</li> <li>• "no": not to draw any total table.</li> </ul>
totaltab_name	The name of the total table, as a single string.
tot	<p>The totals :</p> <ul style="list-style-type: none"> <li>• c("col", "row") or "both" : by default, both total rows and total columns.</li> <li>• "row": only total rows.</li> <li>• "col": only total column.</li> <li>• "no": remove all totals (after calculations if needed).</li> </ul>

total_names	The names of the totals, as a character vector of length one or two. Use syntax of type <code>c("Total row", "Total column")</code> to set different names for rows and cols.
chi2	Set to TRUE to calculate Chi2 summaries with <code>tab_chi2</code> . Useful to print meta-data, and to color cells based on their contribution to variance ( <code>color = "contrib"</code> ). Automatically added if needed for color.
ci	The type of confidence intervals to calculate, passed to <code>tab_ci</code> (automatically added if needed for color). <ul style="list-style-type: none"> <li>• "cell": absolute confidence intervals of cells percentages.</li> <li>• "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when <code>diff = "first"</code>).</li> <li>• "auto": <code>ci = "diff"</code> for means and row/col percentages, <code>ci = "cell"</code> for frequencies ("all", "all_tabs").</li> </ul> <p>By default, for percentages, with <code>ci = "cell"</code> Wilson's method is used, and with <code>ci = "diff"</code> Wald's method along Agresti and Caffo's adjustment. Means use classic method. This can be changed in <code>tab_ci</code>.</p>
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 <code>tab_prepare</code> .
rare_to_other	When set to TRUE, levels with less count than <code>n_min</code> 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 <code>dplyr::filter</code> 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 <code>tibble::tribble</code> , 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:

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

# Colors to help the user read the table:
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:

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

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

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

# Historical differences:

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

# Same differences, minus their confidence intervals:
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:
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 :

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:
tab(data, race, marital, year, pct = "row") %>%

```

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

---

tab\_chi2

*Add Chi2 summaries to a [tab](#)*


---

## Description

Add Chi2 summaries to a [tab](#)

## Usage

```
tab_chi2(
  tabs,
  calc = c("ctr", "p", "var", "counts"),
  comp = NULL,
  color = c("no", "auto", "all", "all_pct")
)
```

## Arguments

- |       |   |
|-------|---|
| tabs  | A tibble of class <code>tab</code> , made with <a href="#">tab_plain</a> or <a href="#">tab_many</a> .  |
| 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 <code>c("ctr", "p", "var", "counts")</code> .  |
| comp  | Comparison level. When <code>tab_vars</code> are present, should the contributions to variance be calculated for each subtable/group (by default, <code>comp = "tab"</code> ) ? Should they be calculated for the whole table ( <code>comp = "all"</code> ) ? <code>comp</code> must be set once and for all the first time you use <a href="#">tab_plain</a> , <a href="#">tab_num</a> or <a href="#">tab_chi2</a> with rows, or <a href="#">tab_ci</a> .                      |
| color | The type of colors to print, as a single string. <ul style="list-style-type: none"> <li>• "no": by default, no colors are printed</li> <li>• "all": color all cells based on their contribution to variance (except for mean columns, from numeric variables)</li> <li>• "all_pct": color all percentages cells based on their contribution to variance</li> <li>• "auto": only color columns with counts, <code>pct = "all"</code> or <code>pct = "all_tabs"</code></li> </ul> |

## Value

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

## Examples

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

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")) %>%
  tab_chi2(calc = c("p", "ctr"), color = TRUE)
```

---

tab_ci	<i>Add confidence intervals to a <a href="#">tab</a></i>
--------	--

---

## Description

Add confidence intervals to a [tab](#)

## Usage

```
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 <code>tab</code> made with <a href="#">tab_plain</a> or <a href="#">tab_many</a> .
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 <code>diff</code> is not "tot" in <a href="#">tab_plain</a> or <a href="#">tab_num</a> ). By default, "diff" ci are calculated for means and row and col percentages, "cell" ci for frequencies ("all", "all_tabs").
comp	Comparison level. When <code>tab_vars</code> are present, should the contributions to variance be calculated for each subtable/group (by default, <code>comp = "tab"</code> )? Should they be calculated for the whole table ( <code>comp = "all"</code> )? <code>comp</code> must be set once and for all the first time you use <a href="#">tab_plain</a> , <a href="#">tab_num</a> or <a href="#">tab_chi2</a> with rows, or <a href="#">tab_ci</a> .
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. <ul style="list-style-type: none"> <li>• "no": by default, no colors are printed</li> <li>• "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</li> <li>• "after_ci": idem, but cut off the confidence interval from the difference</li> </ul>
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 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 <a href="#">BinomCI</a> .
method_diff	Character string specifying which method to use with percentages for ci = "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 <a href="#">BinomDiffCI</a> .

### Value

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

### Examples

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

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**

```
tab_kable(
  tabs,
  theme = c("light", "dark"),
  color_type = NULL,
  html_24_bit = NULL,
  tooltips = TRUE,
  popover = NULL,
  ...
)
```

**Arguments**

tabs	A table made with <a href="#">tab</a> or <a href="#">tab_many</a> .
theme	By default, a white table with black text, Set to "dark" for a black table with white text.
color_type	Set to "text" to color the text, "bg" to color the background. By default it takes <code>getOption("tabxplor.color_style_type")</code> .
html_24_bit	Should specific 24bits colors palettes be used ? Default to <code>getOption("tabxplor.color_html_24_bit")</code>
tooltips	By default, html tooltips are used to display additional informations at mouse hover. Set to FALSE to discard.
popover	By default, takes <code>getOption("tabxplor.kable_popover")</code> . When FALSE, html tooltips are of the base kind : they can't be used with floating table of content in pkgmarkdown documents. Set to TRUE to use <b>kableExtra</b> 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="popover"]').popover(); }); &lt;/script&gt;</code> You can then use a css chunk in rmarkdown to change popovers colors.
...	Other arguments to pass to <a href="#">knitr::kable</a> and <a href="#">kableExtra::kable_styling</a> .

**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**

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

---

`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.

### 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  
)
```

```

tab_get_vars(tabs, vars = c("row_var", "col_vars", "tab_vars"))

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	A data frame.
row_vars	The row variable, which will be printed with one level per line. If numeric, it will be converted to factor. If more than one row_var if provided, a different table is made for each of them.
col_vars	<a href="#">&lt;tidy-select&gt;</a> 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 <code>col_vars = c(col_var1, col_var2, ...)</code> .
tab_vars	<a href="#">&lt;tidy-select&gt;</a> One subtable is made for each combination of levels of the tab variables. To pass many variables you may use syntax <code>tab_vars = c(tab_var1, tab_var2, ...)</code> . All tab variables are converted to factor. Leave empty to make a simple table.
wt	A weight variable, of class numeric. Leave empty for unweighted results.
levels	The levels of col_vars to keep (for more complex selections use <a href="#">dplyr::select</a> ). The argument is vectorised over col_vars. <ul style="list-style-type: none"> <li>• "all": by default, all levels are kept.</li> <li>• "first": only keep the first level of each col_vars</li> </ul>
na	The policy to adopt with missing values. It must be a single string. <ul style="list-style-type: none"> <li>• na = "keep": by default, prints NA's as explicit "NA" level.</li> <li>• 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).</li> </ul>

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	The number of digits to print, as a single integer, or an integer vector the same length as col_vars. The argument is vectorised over col_vars.
totaltab	The total table, if there are subtables/groups (i.e. when tab_vars is provided). Vectorised over row_vars. <ul style="list-style-type: none"> <li>• "line": by default, add a general total line (necessary for calculations with comp = "all")</li> <li>• "table": add a complete total table (i.e. row_var by col_vars without tab_vars).</li> <li>• "no": not to draw any total table.</li> </ul>
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. <ul style="list-style-type: none"> <li>• "last": by default, only prints a total column for the last column variable (of class factor, not numeric).</li> <li>• "each": print a total column for each column variable.</li> <li>• "no": remove all total columns (after calculations if needed).</li> </ul>
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 : <ul style="list-style-type: none"> <li>• "row": row percentages.</li> <li>• "col": column percentages.</li> <li>• "all": frequencies for each subtable/group, if there is tab_vars.</li> <li>• "all_tabs": frequencies for the whole (set of) table(s).</li> </ul> <p>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"))</p>
diff	The reference cell to calculate differences (used to print colors). Vectorised over row_vars. <ul style="list-style-type: none"> <li>• "tot": by default, cells differences from total rows are calculated with pct = "row", and cells differences from total columns with pct = "col".</li> <li>• "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).</li> <li>• n: when diff is an integer, the nth row (or column) is used for comparison.</li> <li>• "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.</li> <li>• "no": not use diffs.</li> </ul>



comp	<p>The comparison level : by subtables/groups, or for the whole table. Vectorised over row_vars.</p> <ul style="list-style-type: none"> <li>• "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.</li> <li>• "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.</li> </ul>
chi2	<p>Set to TRUE to calculate Chi2 summaries with <code>tab_chi2</code>. Useful to print meta-data, and to color cells based on their contribution to variance (color = "contrib"). Vectorised over row_vars.</p>
ci	<p>The type of confidence intervals to calculate, passed to <code>tab_ci</code>. Vectorised over row_vars.</p> <ul style="list-style-type: none"> <li>• "cell": absolute confidence intervals of cells percentages.</li> <li>• "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when diff = "first").</li> <li>• "auto": ci = "diff" for means and row/col percentages, ci = "cell" for frequencies ("all", "all_tabs").</li> </ul> <p>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 <code>tab_ci</code>.</p>
conf_level	<p>The confidence level, as a single numeric between 0 and 1. Default to 0.95 (95%).</p>
method_cell	<p>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 <a href="#">BinomCI</a>.</p>
method_diff	<p>Character string specifying which method to use with percentages for ci = "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 <a href="#">BinomDiffCI</a>.</p>
color	<p>The type of colors to print, as a single string. Vectorised over row_vars.</p> <ul style="list-style-type: none"> <li>• "no": by default, no colors are printed.</li> <li>• "diff": color percentages and means based on cells differences from totals (or from first cells when diff = "first").</li> <li>• "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.</li> <li>• "after_ci": idem, but cut off the confidence interval from the difference first.</li> <li>• "contrib": color cells based on their contribution to variance (except mean columns, from numeric variables).</li> </ul>

- "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.
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 <code>tab_prepare</code> .
rare_to_other	When set to TRUE, levels with less count than <code>n_min</code> 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 <code>dplyr::filter</code> 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 <code>tibble::tribble</code> , to use different filters for similar tables or simply make the field of observation more visible into the code.
tabs	A tibble of class <code>tab</code> , made with <code>tab</code> , <code>tab_many</code> or <code>tab_plain</code> .
vars	In <code>tab_get_vars</code> , a character vector containing the wanted vars names: "row_var", "col_vars" or "tab_vars".
x	A object to test with <code>is_tab</code> .
type	Default to "positive", which just print breaks for positive spreads. Set to all to get breaks for negative spreads as well.
theme	For <code>set_color_style</code> and <code>get_color_style</code> , 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 <code>getOption("tabxplor.color_html_24_bit")</code>
custom_palette	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 <code>options("tabxplor.color_style")</code> . To discard, relaunch the function with <code>custom_palette = NULL</code> .
mode	By default, <code>get_color_style</code> returns a list of <b>crayon</b> coloring functions. Set to "color_code" to return html color codes.
pct_breaks	If they are to be changed, the breaks used for percentages. Default to <code>c(0.05, 0.1, 0.2, 0.3)</code> : 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 <code>color = "after_ci"</code> , the first break is subtracted from all breaks (default becomes <code>c(0, 0.05, 0.15, 0.25)</code> : +0%, +5%, +15%, +25%).
mean_breaks	If they are to be changed, the breaks used for means. Default to <code>c(1.15, 1.5, 2, 4)</code> : 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 <code>color = "after_ci"</code> , the first break is divided from all breaks (default becomes <code>c(1, 1.3, 1.7, 3.5)</code> ).
contrib_breaks	If they are to be changed, the breaks used for contributions to variance. Default to <code>c(1, 2, 5, 10)</code> : 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.

brk When missing, return all color breaks. Specify to return a given color break, among "pct", "mean", "contrib", "pct\_ci" and "mean\_ci".

## Value

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 :
```

```
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:

library(purrr)
library(tibble)
pmap(
  tribble(
    ~row_var, ~col_vars      , ~pct , ~filter      , ~subtext
    "race"   , "marital"    , "row", NULL      , "Source: GSS 2000-2014",
    "relig"  , c("race", "age"), "row", "year %in% 2000:2010", "Source: GSS 2000-2010",
    NA_character_ , "race"    , "no" , NULL      , "Source: GSS 2000-2014",
  ),
  .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

```

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) : <ul style="list-style-type: none"> <li>• "tot": by default, cells differences from total rows are calculated with pct = "row", and cells differences from total columns with pct = "col".</li> <li>• "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).</li> <li>• "no": not use diffs to gain calculation time.</li> </ul>
ci	The type of confidence intervals to calculate, passed to <code>tab_ci</code> (automatically added if needed for color). <ul style="list-style-type: none"> <li>• "cell": absolute confidence intervals of cells percentages.</li> <li>• "diff": confidence intervals of the difference between a cell and the relative total cell (or relative first cell when diff = "first").</li> <li>• "auto": ci = "diff" for means and row/col percentages, ci = "cell" for frequencies ("all", "all_tabs").</li> </ul> <p>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 <code>tab_ci</code>.</p>
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 <code>tab_plain</code> , <code>tab_num</code> or <code>tab_chi2</code> with rows, or <code>tab_ci</code> .
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. <ul style="list-style-type: none"> <li>• "keep": by default, NA's of row and tab variables are printed as an explicit "NA" level.</li> </ul>

- "drop": remove NA's in row and tab variables.

NAs 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.

<code>totaltab</code>	The total table, if there are subtables/groups (i.e. when <code>tab_vars</code> is provided) : <ul style="list-style-type: none"> <li>• "line": by default, add a general total line (necessary for calculations with <code>comp = "all"</code>)</li> <li>• "table": add a complete total table (i.e. <code>row_var</code> by <code>col_vars</code> without <code>tab_vars</code>).</li> <li>• "no": not to draw any total table.</li> </ul>
<code>totaltab_name</code>	The name of the total table, as a single string.
<code>tot</code>	The totals : <ul style="list-style-type: none"> <li>• <code>c("col", "row")</code> or "both" : by default, both total rows and total columns.</li> <li>• "row": only total rows.</li> <li>• "col": only total column.</li> <li>• "no": remove all totals (after calculations if needed).</li> </ul>
<code>total_names</code>	The names of the totals, as a character vector of length one or two. Use syntax of type <code>c("Total row", "Total column")</code> to set different names for rows and cols.
<code>subtext</code>	A character vector to print rows of legend under the table.
<code>num</code>	Set to TRUE to obtain a table with normal numeric vectors (not <code>fmt</code> ).
<code>df</code>	Set to TRUE to obtain a plain data.frame (not a tibble), with normal numeric vectors (not <code>fmt</code> ). Useful, for example, to pass the table to correspondence analysis with <b>FactoMineR</b> .

### Value

A tibble of class `tabexplor_tab`. If ... (`tab_vars`) are provided, a `tab` of class `tabexplor_grouped_tab`. All non-text columns are `fmt` vectors of class `tabexplor_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	<i>Add percentages and diffs to a <a href="#">tab</a></i>
---------	---

---

## 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 <code>tab</code> made with <a href="#">tab_plain</a> or <a href="#">tab_many</a> .
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 <code>tab_vars</code> 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 <code>col_vars</code> .
diff	By default, with <code>pct = "row"</code> , differences from total rows are calculated, and with <code>pct = "col"</code> differences from total columns. Set to <code>diff = "first"</code> to calculate differences with the first cell of the row/col (useful to color temporal developments). When not using diffs for colors, set to <code>diff = "no"</code> to gain calculation time. Diffs are also calculated for mean columns (made from numeric variables).
comp	Comparison level. When <code>tab_vars</code> are present, should the row differences be calculated for each subtable/group (by default <code>comp = "tab"</code> : comparison of each cell to the relative total row) ? Should they be calculated for the whole table ( <code>comp = "all"</code> : comparison of each cell to the total row of the total table) ? When <code>comp = "all"</code> and <code>diff = "first"</code> , cells are compared to the first cell of the total table instead. This parameter doesn't affect column percentages. <code>comp</code> must be set once and for all the first time you use <a href="#">tab_chi2</a> , <a href="#">tab_pct</a> with rows, or <a href="#">tab_ci</a> .
color	Set to TRUE to color the resulting <code>tab</code> 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.

---

<code>tab_plain</code>	<i>Plain single cross-table</i>
------------------------	---------------------------------

---

**Description**

Plain single cross-table

**Usage**

```
tab_plain(
  data,
  row_var,
  col_var,
  tab_vars,
  wt,
  pct = "no",
  diff = "tot",
  comp = "tab",
  color = TRUE,
  digits = 0,
  na = "keep",
  subtext = "",
  totaltab = "line",
  totaltab_name = "Ensemble",
  tot = NULL,
  total_names = "Total",
  num = FALSE,
  df = FALSE
)
```

**Arguments**

<code>data</code>	A data frame.
<code>row_var, col_var</code>	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 <code>tab_num</code> .
<code>tab_vars</code>	<code>&lt;tidy-select&gt;</code> 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.
<code>wt</code>	A weight variable, of class numeric. Leave empty for unweighted results.
<code>pct</code>	The type of percentages to calculate :



	<ul style="list-style-type: none"> <li>• "row": row percentages.</li> <li>• "col": column percentages.</li> <li>• "all": frequencies for each subtable/group, if there is tab_vars.</li> <li>• "all_tabs": frequencies for the whole (set of) table(s).</li> </ul>
diff	<p>The reference cell to calculate differences (used to print colors) :</p> <ul style="list-style-type: none"> <li>• "tot": by default, cells differences from total rows are calculated with pct = "row", and cells differences from total columns with pct = "col".</li> <li>• "first": calculate cells differences from the first cell of the row or column (useful to color temporal developments).</li> <li>• "no": not use diffs to gain calculation time.</li> </ul>
comp	<p>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 <code>tab_plain</code>, <code>tab_num</code> or <code>tab_chi2</code> with rows, or <code>tab_ci</code>.</p>
color	<p>TRUE print the color percentages and means based on cells differences from totals or reference cell, as provided by <code>diff</code>. Default to FALSE, no colors.</p>
digits	<p>The number of digits to print, as a single integer.</p>
na	<p>The policy to adopt with missing values, as a single string.</p> <ul style="list-style-type: none"> <li>• "keep": by default, NA's of row, col and tab variables are printed as explicit "NA" level.</li> <li>• "drop": removes NA of row, col and tab variables.</li> </ul>
subtext	<p>A character vector to print rows of legend under the table.</p>
totaltab	<p>The total table, if there are subtables/groups (i.e. when tab_vars is provided) :</p> <ul style="list-style-type: none"> <li>• "line": by default, add a general total line (necessary for calculations with comp = "all")</li> <li>• "table": add a complete total table (i.e. row_var by col_vars without tab_vars).</li> <li>• "no": not to draw any total table.</li> </ul>
totaltab_name	<p>The name of the total table, as a single string.</p>
tot	<p>The totals :</p> <ul style="list-style-type: none"> <li>• c("col", "row") or "both" : by default, both total rows and total columns.</li> <li>• "row": only total rows.</li> <li>• "col": only total column.</li> <li>• "no": remove all totals (after calculations if needed).</li> </ul>
total_names	<p>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.</p>
num	<p>Set to TRUE to obtain a table with normal numeric vectors (not fmt).</p>
df	<p>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 <b>FactoMineR</b>.</p>

**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**

```
# 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")
```

---

<code>tab_prepare</code>	<i>Prepare data for <code>tab_plain</code>.</i>
--------------------------	---

---

**Description**

Prepare data for `tab_plain`.

**Usage**

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

**Arguments**

<code>data</code>	A dataframe.
<code>...</code>	Variables then to be passed in <code>tab_plain</code> .
<code>na_drop_all</code>	<code>&lt;tidy-select&gt;</code> Removes all observation with a NA in any of the chosen variables.
<code>cleannames</code>	Set to TRUE to clean levels names, by removing prefix numbers like "1-", and text in parentheses.
<code>rare_to_other</code>	When set to TRUE, levels with less count than <code>n_min</code> 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 data.frame.

**Examples**

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

---

tab_spread	<i>Spread a tab, passing a tab variable to column</i>
------------	---

---

**Description**

Spread a tab, passing a tab variable to column

**Usage**

```
tab_spread(
  tabs,
  spread_vars,
  names_prefix,
  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**

```

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)

```

---

tab_tot	<i>Add totals to a <a href="#">tab</a></i>
---------	--

---

**Description**

Add totals to a [tab](#)

**Usage**

```

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

```

**Arguments**

tabs	A tibble of class <code>tab</code> , made with <a href="#">tab_plain</a> or <a href="#">tab_many</a> .
tot	<code>c("col", "row")</code> and <code>"both"</code> print total rows and total columns. Set to <code>"row"</code> or <code>"col"</code> to print only one type. Set to <code>"no"</code> to remove all totals.
name	The names of the totals, as a character vector of length one or two. Use <code>c("Total_row", "Total_column")</code> to set different names for rows and cols.
totcol	<code>"last"</code> only prints a total column for the last factor column variable. Set to <code>"each"</code> to print a total column for each column variable.
data	The original database used to calculate the <code>tab</code> : 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 <a href="#">tab_ci</a> .

**Value**

A tibble of class `tab`. Total rows can then be detected using [is\\_totrow](#), and total columns using [is\\_totcol](#).

**Examples**

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

data %>%
  tab_plain(sex, hair_color) %>%
  tab_tot("col", totcol = "each")
```

---

tab_totaltab	<i>Add total table to a <a href="#">tab</a></i>
--------------	---

---

**Description**

Add total table to a [tab](#)

**Usage**

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

**Arguments**

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

**Value**

A tibble of class `tab`. Rows belonging to the total table can then be detected using [is\\_tottab](#).

**Examples**

```
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")
```

---

tab_xl	<i>Excel output for tabxplor tables, with formatting and colors</i>
--------	---

---

### 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

```
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",
  min_counts = 30,
  hide_near_zero = "auto",
  color_type = "text"
)
```

### Arguments

tabs	A table made with <a href="#">tab</a> , <a href="#">tab_many</a> or <a href="#">tab_plain</a> , 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 <code>link[base:options]{options}</code> to change default directory. Use <code>replace = TRUE</code> to overwrite existing files. Use <code>open = TRUE</code> 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, <code>tab_vars</code> columns are removed to gain space. Set to <code>FALSE</code> to keep them.
colwidth	The standard width for numeric columns, as a number. Default to "auto".
print_ci	By default provided confidence intervals are printed in another table, left to the base table. Set to <code>FALSE</code> to dismiss.
print_color_legend	Should the color legends be printed with the subtexts ?

sheets	The Excel sheets options : <ul style="list-style-type: none"> <li>• "tabs": a new sheet is created for each table</li> <li>• "unique": all tables are on the same sheet</li> <li>• "auto": subsequent tables with the same columns are printed on the same sheets</li> </ul>
min_counts	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 30.
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 background is colored. Set to text to color the text instead.

**Value**

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

**Examples**

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

---

tbl\_format\_body.tabxplor\_tab  
*Table body for class tab*

---

**Description**

Table body for class tab

**Usage**

```
## 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.

---

tbl\_format\_footer.tabxplor\_tab  
*Table footer for class tab*

---

**Description**

Table footer for class tab

**Usage**

```
## 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.

---

tbl\_sum.tabxplor\_grouped\_tab  
*Table headers for class grouped tab*

---

**Description**

Table headers for class grouped tab

**Usage**

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

**Arguments**

x	An object of class tabxplor_tab
...	Other parameters.

**Value**

A table header



---

tbl\_sum.tabxplor\_tab *Table headers for class tab*

---

**Description**

Table headers for class tab

**Usage**

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

**Arguments**

x                    An object of class tabxplor\_tab  
...                   Other parameters.

**Value**

A table header

---

ungroup.tabxplor\_grouped\_tab  
*ungroup method for class tabxplor\_grouped\_tab*

---

**Description**

ungroup method for class tabxplor\_grouped\_tab

**Usage**

```
## 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 'tabxplor_fmt'  
vec_arith.numeric(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

## Methods (by class)

- default: default vec\_arith method for fmt
- tabxplor\_fmt: vec\_arith method for fmt + fmt
- numeric: vec\_arith method for fmt + numeric
- tabxplor\_fmt: vec\_arith method for numeric + fmt
- MISSING: vec\_arith method for -fmt

---

vec\_cast.character.tabxplor\_fmt  
*Convert fmt into character*

---

**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 #' @return 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**

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

**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**

x	A integer vector
to	A fmt vector
...	Other parameter. #' @return 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. #' @return 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.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. #' @return 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
...	Other parameter

### 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
...	Other parameter

**Value**

A double vector

---

```
vec_ptype2.double.tabxplor_fmt
    Find common ptype between double and fmt
```

---

**Description**

Find common ptype between double and fmt

**Usage**

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

**Arguments**

x	A double vector
y	A fmt vector
...	Other parameter. #' @return A fmt vector

---

```
vec_ptype2.integer.tabxplor_fmt
```

*Find common ptype between integer and fmt*

---

### Description

Find common ptype between integer and fmt

### Usage

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

### Arguments

x	An integer vector
y	A fmt vector
...	Other parameter. #' @return 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. #' @return 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**

```
## 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

```
## 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

```
## 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\_fmt      *\$ method for class tabxplor\_fmt*

---

### **Description**

\$ method for class tabxplor\_fmt

### **Usage**

```
## S3 method for class 'tabxplor_fmt'  
x$name
```

### **Arguments**

x                    A tabxplor\_fmt object.  
name                The name of the field to extract.

### **Value**

The relevant field of the tabxplor\_fmt.

# Index

\$. tabxplor\_fmt, 75

arrange, 37, 46

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

dplyr::filter, 40, 50

dplyr::select, 47

dplyr\_col\_modify. tabxplor\_grouped\_tab, 4

dplyr\_reconstruct. tabxplor\_grouped\_tab, 4

dplyr\_row\_slice. tabxplor\_grouped\_tab, 5

fct\_recode\_helper, 5

filter, 37, 46

fmt, 6, 8, 40, 51, 54, 58

fmt\_get\_color\_code, 13

format. tabxplor\_fmt, 13

get\_ci\_type (fmt), 6

get\_ci\_type.data.frame, 14

get\_ci\_type.default, 14

get\_ci\_type. tabxplor\_fmt, 15

get\_col\_var (fmt), 6

get\_col\_var.data.frame, 17

get\_col\_var.default, 17

get\_col\_var. tabxplor\_fmt, 18

get\_color (fmt), 6

get\_color.data.frame, 15

get\_color.default, 16

get\_color. tabxplor\_fmt, 16

get\_color\_breaks (tab\_many), 46

get\_color\_style (tab\_many), 46

get\_comp\_all, 9

get\_comp\_all (fmt), 6

get\_diff\_type (fmt), 6

get\_diff\_type.data.frame, 18

get\_diff\_type.default, 19

get\_diff\_type. tabxplor\_fmt, 19

get\_digits (fmt), 6

get\_num (fmt), 6

get\_type (fmt), 6

get\_type.data.frame, 20

get\_type.default, 20

get\_type. tabxplor\_fmt, 21

group\_by. tabxplor\_tab, 21

is\_fmt (fmt), 6

is\_refcol (fmt), 6

is\_refcol.data.frame, 22

is\_refcol.default, 22

is\_refcol. tabxplor\_fmt, 23

is\_refrow (fmt), 6

is\_refrow.data.frame, 23

is\_refrow.default, 24

is\_refrow. tabxplor\_fmt, 24

is\_tab, 50

is\_tab (tab\_many), 46

is\_totcol, 60

is\_totcol (fmt), 6

is\_totcol.data.frame, 25

is\_totcol.default, 25

is\_totcol. tabxplor\_fmt, 26

is\_totrow, 60

is\_totrow (fmt), 6

is\_totrow.data.frame, 26

is\_totrow.default, 27

is\_totrow. tabxplor\_fmt, 27

is\_tottab, 61

is\_tottab (fmt), 6

is\_tottab.data.frame, 28

is\_tottab.default, 28

is\_tottab.tabxplor\_fmt, 29  
 kableExtra::kable\_styling, 45  
 knitr::kable, 45  
 mutate, 37, 46  
 mutate.tabxplor\_fmt, 29  
 new\_grouped\_tab (new\_tab), 30  
 new\_tab, 30  
 pillar\_shaft.tab\_chi2\_fmt, 31  
 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, 46  
 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), 46  
 set\_color\_style, 62  
 set\_color\_style (tab\_many), 46  
 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, 30, 37, 42, 43, 45, 50, 51, 55, 59–62  
 tab\_chi2, 8, 30, 40, 42, 42, 43, 49, 53, 55, 57  
 tab\_ci, 9, 40, 42, 43, 43, 49, 53, 55, 57, 60, 61  
 tab\_get\_vars (tab\_many), 46  
 tab\_kable, 44  
 tab\_many, 30, 37, 38, 42, 43, 45, 46, 50, 54, 55, 59–62  
 tab\_num, 42, 43, 52, 53, 56, 57  
 tab\_pct, 8, 9, 55, 55  
 tab\_plain, 30, 42, 43, 50, 53, 55, 56, 57–62  
 tab\_prepare, 40, 50, 54, 58  
 tab\_spread, 59  
 tab\_tot, 60  
 tab\_totaltab, 61  
 tab\_xl, 37, 46, 62  
 tbl\_format\_body.tabxplor\_tab, 63  
 tbl\_format\_footer.tabxplor\_tab, 64  
 tbl\_sum.tabxplor\_grouped\_tab, 64  
 tbl\_sum.tabxplor\_tab, 65  
 tibble, 30, 31  
 tibble::tribble, 40, 50  
 tidy-select, 5, 38, 47, 48, 53, 56, 58, 59  
 ungroup.tabxplor\_grouped\_tab, 65  
 vctrs::field, 6  
 vec\_arith.numeric.tabxplor\_fmt  
     (vec\_arith.tabxplor\_fmt), 66  
 vec\_arith.tabxplor\_fmt, 66  
 vec\_cast.character.tabxplor\_fmt, 67  
 vec\_cast.double.tabxplor\_fmt, 67  
 vec\_cast.integer.tabxplor\_fmt, 68  
 vec\_cast.tabxplor\_fmt.double, 68  
 vec\_cast.tabxplor\_fmt.integer, 69  
 vec\_cast.tabxplor\_fmt.tabxplor\_fmt, 69  
 vec\_math.tabxplor\_fmt, 70  
 vec\_proxy\_compare.tabxplor\_fmt, 70  
 vec\_proxy\_equal.tabxplor\_fmt, 71  
 vec\_ptype2.double.tabxplor\_fmt, 71  
 vec\_ptype2.integer.tabxplor\_fmt, 72  
 vec\_ptype2.tabxplor\_fmt.double, 72  
 vec\_ptype2.tabxplor\_fmt.integer, 73  
 vec\_ptype2.tabxplor\_fmt.tabxplor\_fmt,  
     73  
 vec\_ptype\_abbr.tabxplor\_fmt, 74  
 vec\_ptype\_full.tabxplor\_fmt, 74