Introducing pmtables

Author: Metrum Research Group

Date: 2022-04-19

pmtables: v0.4.1.9003

Introduction: This is a simple introduction to the pmtables package for R. I hope this will be useful for those who are new to the package and those who just need a reminder on the syntax.

Contents

1 A simple table: stable

1.1 Syntax

Pass your data.frame into `stable()`

```
stable(data)
```

Other formal arguments include:

- **align** to set column alignment
- **panel** to create groups of rows under a "panel" header
- **span** to group columns under a "spanner" header
- **notes** to create table notes
- **sumrows** to insert summary rows
- **units** that get placed below the corresponding column name
- **drop** to omit certain columns from the table
- **sizes** to set different table size attributes
- **escape_fun** a function to sanitize table items

You can also pass a bunch of other arguments through `...` to further format the table (see `?stable` for details)

1.2 Basics

`stable()` is the name of the workhorse function that is used to turn data.frames into TeX tables. This chapter will introduce the `stable()` function and how to us it to create basic tables.

To illustrate usage and features of `stable()`, we will use the `stdata` data set that comes with pmtables

```
data <- stdata()

head(data)
```

```
. # A tibble: 6 x 9
.   STUDY       DOSE   FORM    N     WT    CRCL  AGE   ALB   SCR
.   <chr>       <chr>  <chr>   <chr> <chr> <chr> <chr> <chr> <chr>
. 1 12-DEMO-001 100 mg tablet  80    71.4  104   33.7  4.20  1.06
. 2 12-DEMO-001 150 mg capsule 16    89.4  122   24.4  4.63  1.12
. 3 12-DEMO-001 150 mg tablet  48    81.7  104   34.4  3.83  0.910
. 4 12-DEMO-001 150 mg troche  16    94.0  93.2  27.4  4.94  1.25
. 5 12-DEMO-001 200 mg tablet  64    67.9  100   27.5  4.25  1.10
. 6 12-DEMO-001 200 mg troche  16    76.6  99.2  22.8  4.54  1.15
```

We can turn this data frame into a TeX table by passing it into `stable()`.

```
out <- stable(data)

head(out, n = 10)
```

```
.  [1] "\\setlength{\\tabcolsep}{5pt} "
.  [2] "\\begin{threeparttable}"
.  [3] "\\renewcommand{\\arraystretch}{1.3}"
.  [4] "\\begin{tabular}[h]{lllllllll}"
.  [5] "\\hline"
.  [6] "STUDY & DOSE & FORM & N & WT & CRCL & AGE & ALB & SCR \\\\"
.  [7] "\\hline"
.  [8] "12-DEMO-001 & 100 mg & tablet & 80 & 71.4 & 104 & 33.7 & 4.20 & 1.06 \\\\"
.  [9] "12-DEMO-001 & 150 mg & capsule & 16 & 89.4 & 122 & 24.4 & 4.63 & 1.12 \\\\"
. [10] "12-DEMO-001 & 150 mg & tablet & 48 & 81.7 & 104 & 34.4 & 3.83 & 0.910 \\\\"
```

Note that we have shown the raw latex code that is generated by `stable()`. That is to say: the output from `stable()` is a character vector of latex code for the table. Note also that this character vector has a special class associated with it: `stable`. That means we can write functions that recognize this character vector as output from `stable()` and we can have those functions process the character vector in special ways.

We can render that table in TeX **in the current Rmarkdown document** by passing the text to `st_asis()`.

```
out %>% st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	troche	36	73.6	103	49.2	4.52	0.930

Remember to only call `st_asis()` when you are rendering tables inline in an Rmd document. If you are sending table code to a TeX report, then you will save them to a file and then include them into your report.

The remaining sections of this chapter will show you how to modify and enhance this output in the more basic ways. We will implement separate chapters for more complicated table manipulations.

1.3 Annotate with file names

pmtables can track and annotate your table with the filenames of the R code that generated the table (`r_file`) as well as the output file where you write the the table `.tex` code (`output_file`).

To have pmtables annotate your table with these file names, pass them in with the `r_file` and `output_file` arguments

```
out <- stable(data, r_file = "tables.R", output_file = "tables.tex")
```

When we look at the rendered table, these names will show up as annotations at the bottom of the table

```
out %>% st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	troche	36	73.6	103	49.2	4.52	0.930

Source code: tables.R
Source file: tables.tex

1.4 Saving your stable

Saving your stable **can** be as easy as sending it into `writeLines()`

```
writeLines(out, con = tempfile(tmpdir = '.', fileext = ".tex"))
```

But remember that we passed in the `output_file` argument to `stable()` and we can use that data to save the table code to the file we named in that argument.

Note that our `stable` object has another attribute now called `stable_file`

```
attributes(out)
```

```
. $class
. [1] "stable"
.
. $stable_file
. [1] "tables.tex"
```

This has the value that we passed in as `output_file`. To save our table to `stable_file`, we call `stable_save()`

```
stable_save(out)
```

There is a `dir` argument to `stable_save()` that we can use to to select the directory where the file will be saved

```
stable_save(out, dir = tempdir())
```

And if you look at the default value for `dir` in `?stable_save`, you'll see that this is associated with an option called `pmtables.dir`; you can set that option to your default output directory and your tables will be saved there until you change that

```
options(pmtables.dir = tempdir())

stable_save(out)
```

1.5 Align columns

Use the `align` argument to align column data to the left, center or right. Use a `cols_*` function to specify the default alignment for all columns

```
tmp <- tibble(AB = 1, CDEFGHIJ = 2, KL = 3)
stable(tmp, align = cols_center()) %>% st_asis()
```

AB	CDEFGHIJ	KL
1	2	3

You can pass in exceptions to the default

```
stable(tmp, align = cols_center(CDEFGHIJ = "r")) %>% st_asis()
```

AB	CDEFGHIJ	KL
1	2	3

Or you can pass an alignment directive and the columns that are bound by that directive

```
stable(tmp, align = cols_center(.l = "AB,KL")) %>% st_asis()
```

AB	CDEFGHIJ	KL
1	2	3

A special directive called `.outer` lets you specify the alignment of the first and last column in the table. For example, this code puts the first column to the left and the last column to the right.

```
stable(tmp, align = cols_center(.outer = "lr")) %>% st_asis()
```

AB	CDEFGHIJ	KL
1	2	3

1.5.1 Fixed column widths

Use `col_ragged(size)` to force a column to be a fixed size.

```
stable(tmp, align = cols_center(AB = col_ragged(2))) %>% st_asis()
```

AB	CDEFGHIJ	KL
1	2	3

By default, the unit is `cm` so that the first column (`AB`) has a width of 2 cm regardless of the contents.

See `cols_align()` help topic for more information and argument descriptions.

1.6 Manipulating columns and names

1.6.1 Rename columns

You can change the name that appears in the rendered table with `cols_rename`

```
data %>%
  slice(1:3) %>%
  stable(cols_rename = c(Age = "AGE", Weight = "WT")) %>%
  st_asis()
```

STUDY	DOSE	FORM	N	Weight	CRCL	Age	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910

Note that the rename syntax follows the tidyselect convention of putting the new name on the left and the old name on the right.

1.6.2 Hide a column name

You can also “erase” the name of a column in the output

```
data %>%
  slice(1:3) %>%
  stable(cols_blank = "WT,ALB,SCR") %>%
  st_asis()
```

STUDY	DOSE	FORM	N		CRCL	AGE		
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910

1.6.3 Don't print any table header information

```
data %>%
  slice(1:3) %>%
  stable(cols_omit = TRUE) %>%
  st_asis()
```

12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910

1.6.4 Unmask column names

In tibbles, you can't have duplicate column names. The `cols_split` argument lets you unmask the names when duplicate names are prefixed with a tag and a delimiter

```
tmp <- tibble(a.A = 1, b.A = 2, c.A = 3)
```

```
tmp %>% stable(cols_split = '.') %>% st_asis()
```

A	A	A
1	2	3

1.6.5 Make column names bold

```
data %>% slice(1:2) %>% stable(cols_bold = TRUE) %>% st_asis()
```

STUDY	**DOSE**	**FORM**	**N**	**WT**	**CRCL**	**AGE**	**ALB**	**SCR**
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12

1.6.6 Drop a column from the table

If we want to prevent a column from appearing in the output table (e.g. `FORM`)

```
head(data)
```

```
. # A tibble: 6 x 9
.   STUDY       DOSE    FORM    N     WT    CRCL  AGE   ALB   SCR
.   <chr>       <chr>   <chr>   <chr> <chr> <chr> <chr> <chr> <chr>
. 1 12-DEMO-001 100 mg tablet  80    71.4  104   33.7  4.20  1.06
. 2 12-DEMO-001 150 mg capsule 16    89.4  122   24.4  4.63  1.12
. 3 12-DEMO-001 150 mg tablet  48    81.7  104   34.4  3.83  0.910
. 4 12-DEMO-001 150 mg troche  16    94.0  93.2  27.4  4.94  1.25
. 5 12-DEMO-001 200 mg tablet  64    67.9  100   27.5  4.25  1.10
. 6 12-DEMO-001 200 mg troche  16    76.6  99.2  22.8  4.54  1.15
```

list the column name as `drop`

```
stable(data, drop = "FORM") %>% st_asis()
```

STUDY	DOSE	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	36	73.6	103	49.2	4.52	0.930

Of course some tidyverse could accomplish the same thing

```
data %>% select(-FORM) %>% stable()
```

1.7 Other customizations

1.7.1 Notes

Arbitrary notes can get added to any table using the `notes` argument.

```
data %>%
  slice(1:3) %>%
  stable(notes = "Showing just the first three rows") %>%
  st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910

Showing just the first three rows

The appearance of the notes can be controlled by calling `noteconf()` and passing the result as `note_config`. See `?tab_notes()` for more details.

1.7.2 Units

pmtables can automatically place units underneath the appropriate column. To do this, generate a list with names that match the column names you want to label with units.

```
u <- list(
  WT = "kg", CRCL = "ml/min", AGE = "year", ALB = "g/dL",
  SCR = "mg\\%"
) %>% map(~paste0("(", .x, ")"))
```

Then pass that list as `units` to `stable()`

```
stable(data, units = u) %>% st_asis()
```

STUDY	DOSE	FORM	N	WT (kg)	CRCL (ml/min)	AGE (year)	ALB (g/dL)	SCR (mg%)
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	troche	36	73.6	103	49.2	4.52	0.930

1.7.3 Multi-line column headers

If the column header is long, you can break it across multiple lines. By default, use `...` in the column name

```
tibble(`First line ... Second line` = 123456789) %>%
  stable() %>% st_asis()
```

First line Second line
123456789

The break can be introduced through the rename mechanism

```
tibble(a = 1) %>%
  stable(cols_rename =  c(`First ... Second` = "a")) %>%
  st_asis()
```

First Second
1

Look at the `?tab_cols` help topic for the `cols_break` argument; this lets you change the character sequence used for the break.

1.7.4 Insert horizontal lines

Pass `hlines_at` to insert horizontal lines above specific rows. This can be either logical vector with the same length as the number of rows in the table or a vector of integers.

```
stable(stdata(), hline_at = c(3,5)) %>% st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	troche	36	73.6	103	49.2	4.52	0.930

or

```
stable(stdata(), hline_at = data$FORM == "tablet") %>% st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	troche	36	73.6	103	49.2	4.52	0.930

Pass `hlines_from` to derive hline locations based on non-repeating values in a table column. Notice how this behaves.

```
stable(stdata(), hline_from = "DOSE") %>% st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	troche	36	73.6	103	49.2	4.52	0.930

See the `?tab_hlines` help topic for more info. See also `st_hline()` for the pipe equivalent with additional feature.

1.7.5 Clear replicate values

You can create groups in a table by "clearing" replicate values

```
stable(stdata(), clear_reps = "STUDY") %>% st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
	75 mg	troche	36	73.6	103	49.2	4.52	0.930

This can be combined with an hline

```
stable(stdata(), clear_reps = "STUDY", hline_from = "STUDY") %>%
  st_asis()
```

STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
	75 mg	troche	36	73.6	103	49.2	4.52	0.930

See `?tab_clear_reps` for other options, including an option for clearing based on several grouping variables.

2 Group table rows with panel

2.1 Syntax

To panel a table by `STUDY`

```
stable(stdata(), panel = "STUDY")
```

To set a prefix for the panel header:

```
stable(stdata(), panel = as.panel("STUDY", prefix = "Study: "))
```

2.2 Basics

Paneling your table is a way to group sets of rows together into a "panel" with a panel header rendered in bold font. For example, we can panel a table of `mtcars` by `carb`. We will be working with an abbreviated version of `mtcars`:

```
smcars
```

```
.                          name  mpg cyl  disp  hp drat    wt  qsec vs am gear
. Datsun 710          Datsun 710 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4
. Hornet 4 Drive Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3
. Valiant                Valiant 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3
. Fiat 128              Fiat 128 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4
. Toyota Corolla Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4
. Toyota Corona   Toyota Corona 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3
. Fiat X1-9           Fiat X1-9 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4
. Merc 240D           Merc 240D 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4
. Merc 230             Merc 230 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4
. Honda Civic       Honda Civic 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4
.                 carb
. Datsun 710         1
. Hornet 4 Drive     1
. Valiant            1
. Fiat 128           1
. Toyota Corolla     1
. Toyota Corona      1
. Fiat X1-9          1
. Merc 240D          2
. Merc 230           2
. Honda Civic        2
```

Then we pass into `stable()` and name the paneling column:

```
smcars %>% stable(panel = "carb") %>% st_asis()
```

name	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
1										
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3
Fiat 128	32.4	4	78.7	66	4.08	2.2	19.47	1	1	4
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.9	1	1	4
Toyota Corona	21.5	4	120.1	97	3.7	2.465	20.01	1	0	3
Fiat X1-9	27.3	4	79	66	4.08	1.935	18.9	1	1	4
2										
Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1	0	4
Merc 230	22.8	4	140.8	95	3.92	3.15	22.9	1	0	4
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4

Now, all of the `carb==1` rows are grouped with the heading `1` and similarly the `carb==2` rows are grouped with the heading `2` in bold.

This is ok, but a more informative heading would be helpful. To do this, we'll call `as.panel()` to both name the panel column and set some options:

```
smcars %>% stable(panel = as.panel("carb", prefix = "carb: ")) %>% st_asis()
```

name	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
carb: 1										
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3
Fiat 128	32.4	4	78.7	66	4.08	2.2	19.47	1	1	4
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.9	1	1	4
Toyota Corona	21.5	4	120.1	97	3.7	2.465	20.01	1	0	3
Fiat X1-9	27.3	4	79	66	4.08	1.935	18.9	1	1	4
carb: 2										
Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1	0	4
Merc 230	22.8	4	140.8	95	3.92	3.15	22.9	1	0	4
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4

Note that the prefix is completely specified by the user (including any spaces or a colon.

2.3 panel: additional customization

See the `?rowpanel` help topic for arguments to `rowpanel()` that can be passed to customize the panel. Some of the customizations include

1. add a prefix to the panel title

2. skip making panels for certain data in the panel column
3. make the panel title bold
4. make the panel title italics
5. opt out from drawing `hline` above panels
6. jut the panel titles so that the rows under the panel header are indented (available starting with version 0.4.1)

2.3.1 `jut`

We recommend an value more than 1, otherwise the table looks mis-aligned rather than creating offset between panel row and underlying data.

```
stable(
  stdata(),
  panel = as.panel("STUDY", jut = 1)
) %>% st_asis()
```

DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001							
100 mg	tablet	80	71.4	104	33.7	4.20	1.06
150 mg	capsule	16	89.4	122	24.4	4.63	1.12
150 mg	tablet	48	81.7	104	34.4	3.83	0.910
150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
200 mg	tablet	64	67.9	100	27.5	4.25	1.10
200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002							
100 mg	capsule	36	61.3	113	38.3	4.04	1.28
100 mg	tablet	324	77.6	106	29.9	4.31	0.981
50 mg	capsule	36	74.1	112	37.1	4.44	0.900
50 mg	tablet	324	71.2	106	34.1	4.63	0.868
75 mg	capsule	36	72.4	105	38.2	3.89	0.900
75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
75 mg	troche	36	73.6	103	49.2	4.52	0.930

This feature requires pmtables 0.4.1 or greater.

2.4 panel: important points

1. Most of the time, the data frame should be sorted by the `panel` column
2. pmtables creates panels by non-repeating values in the panel column; there will be an error if duplicate panel names are found and this can be overridden by passing `duplicates_ok` to `as.panel()`.

3 Group table columns with spanners

3.1 Syntax

Set the `span` argument to the output of `as.span()`. The key arguments for `as.span()` are the spanner title and the names of the columns over which you want the spanner to run

```
stable(stdata(), span = as.span("Covariates", WT:SCR))
```

The equivalent pipe syntax is

```
st_new(stdata()) %>% st_span("Covariates", WT:SCR)
```

3.2 Basics

A column spanner puts a horizontal line over a sequence of column names and places a title above that line forming a column group.

As a trivial example:

```
data <-
  tibble(
    Tariffville = "06081",  Connecticut= "CT",
    Minnesota = "MN", Minneapolis = "55455"
  )

data %>% stable(span = as.span("States", Connecticut:Minnesota)) %>%  st_asis()
```

	States		
Tariffville	Connecticut	Minnesota	Minneapolis
06081	CT	MN	55455

3.3 Multiple spanners

Multiple spanners can be added to a table by specifying the `level` for any spanner that you want to be placed above the lowest level spanner. For example,

```
sp <- list(
  as.span("States", Connecticut:Minnesota),
  as.span("Important Locations", Tariffville:Minneapolis, level = 2)
)
data %>% stable(span = sp) %>% st_asis()
```

	Important Locations		
	States		
Tariffville	Connecticut	Minnesota	Minneapolis
06081	CT	MN	55455

Note that to specify multiple spanners, we pass a list of span objects. I've simplified the code a bit here by creating that list as a standalone object and then passing the whole list as span.

3.3.1 Using pipe syntax

For problems like this, it might be preferable to use the pipe syntax

```
data %>%
  st_new() %>%
  st_span("States", Connecticut:Minnesota) %>%
  st_span("Important Locations", Tariffville:Minneapolis, level = 2) %>%
  stable() %>%
  st_asis()
```

Important Locations			
	States		
Tariffville	Connecticut	Minnesota	Minneapolis
06081	CT	MN	55455

3.4 Breaking span title

We can make the title of the span break across multiple lines by using . . .

```
stable(data, span = as.span("Important ... States", Connecticut:Minnesota)) %>%
  st_asis()
```

Important States			
Tariffville	Connecticut	Minnesota	Minneapolis
06081	CT	MN	55455

3.5 Aligning span title

Beginning with version 0.4.1, the span title can be left or right justified in addition to the default centering

```
stable(
  stdata(),
  span = as.span("Covariates", WT:SCR, align = 'l')
) %>% st_asis()
```

				Covariates				
STUDY	DOSE	FORM	N	WT	CRCL	AGE	ALB	SCR
12-DEMO-001	100 mg	tablet	80	71.4	104	33.7	4.20	1.06
12-DEMO-001	150 mg	capsule	16	89.4	122	24.4	4.63	1.12
12-DEMO-001	150 mg	tablet	48	81.7	104	34.4	3.83	0.910
12-DEMO-001	150 mg	troche	16	94.0	93.2	27.4	4.94	1.25
12-DEMO-001	200 mg	tablet	64	67.9	100	27.5	4.25	1.10
12-DEMO-001	200 mg	troche	16	76.6	99.2	22.8	4.54	1.15
12-DEMO-002	100 mg	capsule	36	61.3	113	38.3	4.04	1.28
12-DEMO-002	100 mg	tablet	324	77.6	106	29.9	4.31	0.981
12-DEMO-002	50 mg	capsule	36	74.1	112	37.1	4.44	0.900
12-DEMO-002	50 mg	tablet	324	71.2	106	34.1	4.63	0.868
12-DEMO-002	75 mg	capsule	36	72.4	105	38.2	3.89	0.900
12-DEMO-002	75 mg	tablet	288	71.6	98.9	34.2	4.49	0.991
12-DEMO-002	75 mg	troche	36	73.6	103	49.2	4.52	0.930

3.6 Span created by splitting column names

`colsplit()` is a way to create column spanners which are encoded into the column names of the input data frame; the names are split on a separator character (like `.`) and either the left or right side are taken as the title and the other is taken as the column name.

Consider this data

```
.   A.first A.second B.third B.fourth
. 1       1        2       3        4
```

Notice the natural grouping between `A.first` and `A.second`; we want `first` and `second` grouped together with the title `A`. Similar setup for `third` and `fourth` under the title `B`.

We can make table with spanners by passing a call to `colsplit()` as `span_split`

```
dd  %>% stable(span_split = colsplit(sep = '.')) %>% st_asis()
```

A		B	
first	second	third	fourth
1	2	3	4

3.6.1 pivot_longer workflow

This is pattern is convenient when summarizing data in a long format. To to demonstrate, we'll calculate summary statistics for `WT` and `AGE` by `STUDY`

```
summ <-
  pmt_first %>%
  pivot_longer(cols = c("WT", "AGE")) %>%
  filter(!is.na(value)) %>%
  group_by(STUDYf, name) %>%
  summarise(Mean = mean(value), Sd = sd(value), N = n(), .groups = "drop") %>%
  mutate(across(Mean:N, sig)) %>% mutate(across(Mean:N, as.character))

summ
```

```
. # A tibble: 8 x 5
.   STUDYf      name  Mean  Sd    N
.   <fct>       <chr> <chr> <chr> <chr>
. 1 12-DEMO-001 AGE   32.0  9.19  30
. 2 12-DEMO-001 WT    72.2  14.3  29
. 3 12-DEMO-002 AGE   35.0  8.20  50
. 4 12-DEMO-002 WT    72.4  11.5  49
. 5 11-DEMO-005 AGE   32.8  8.48  40
. 6 11-DEMO-005 WT    68.9  14.5  39
. 7 13-DEMO-001 AGE   34.2  9.67  40
. 8 13-DEMO-001 WT    69.4  11.6  40
```

Now take 2 (or 3) more steps to get the table in the right shape to feed into `stable()`. **First**, pivot this `longer` using the summary stat name

```
long <- pivot_longer(summ, cols = c("Mean", "Sd", "N"), names_to = "stat")
head(long)
```

```
. # A tibble: 6 x 4
.   STUDYf       name  stat  value
.   <fct>        <chr> <chr> <chr>
. 1 12-DEMO-001 AGE   Mean  32.0
. 2 12-DEMO-001 AGE   Sd    9.19
. 3 12-DEMO-001 AGE   N     30
. 4 12-DEMO-001 WT    Mean  72.2
. 5 12-DEMO-001 WT    Sd    14.3
. 6 12-DEMO-001 WT    N     29
```

Second, we'll make `name` more appealing / informative

```
long <- mutate(long, name = recode(name, WT = "Weight (kg)", AGE = "Age (years)"))
```

Third, pivot this `wider` using the covariate `name` and `stat`

```
wide <- pivot_wider(long, names_from = c("name", "stat"), names_sep = "---")
wide
```

```
. # A tibble: 4 x 7
.   STUDYf     `Age (years)---~` `Age (years)---~` `Age (years)---~` `Weight (kg)---~`
.   <fct>      <chr>             <chr>             <chr>             <chr>
. 1 12-DEMO-0~ 32.0              9.19              30                72.2
. 2 12-DEMO-0~ 35.0              8.20              50                72.4
. 3 11-DEMO-0~ 32.8              8.48              40                68.9
. 4 13-DEMO-0~ 34.2              9.67              40                69.4
. # ... with 2 more variables: `Weight (kg)---Sd` <chr>, `Weight (kg)---N` <chr>
```

Now we have column names set up to create the spanners

```
wide %>% stable(span_split = colsplit(sep = "---")) %>% st_asis()
```

	Age (years)			Weight (kg)		
STUDYf	Mean	Sd	N	Mean	Sd	N
12-DEMO-001	32.0	9.19	30	72.2	14.3	29
12-DEMO-002	35.0	8.20	50	72.4	11.5	49
11-DEMO-005	32.8	8.48	40	68.9	14.5	39
13-DEMO-001	34.2	9.67	40	69.4	11.6	40

This workflow takes several steps to complete, but once you identify the pattern it can be just an extra step or two beyond what you're already doing to get a nice table.

4 Tables that span multiple pages: longtable

4.1 Syntax

To create a long table from a data frame

```
stable_long(stdata())
```

To create a long table from pipeline

```
st_new(data) %>% stable_long()
```

To create a long table from `pmtable`

```
pt_cont_long(data, cols = "WT,EGFR") %>% stable_long()
```

4.2 Basics

You can create `longtables` that span multiple pages of your pdf document. Tables using the`longtable` environment are very different than the the basic table from `stable()` which are built using tabular environment.

4.3 Inserting longtable into your latex document

Once you have written your long table out to a file, you can source it into your latex document with a simple input command

```
\input{my-table.tex}
```

IMPORTANT

- Do not wrap the input in `\begin{table}` / `\end{table}`; the table will not show up properly that way
- Do not include any `\caption{...}` statement; the caption must be provided in a special way (see below)

4.4 Writing a caption

Longtable is different than regular tabular table in that the caption (and label) need to be included in the longtable environment. This means that you have to enter this information **when you create the table**.

4.4.1 Method 1: in the R script

`stable_long()` has an argument called `lt_cap_text` that will allow you to provide the text for the caption. This text must be passed to the `stable_long()` call (or `st_make()`)

```
out <- stable_long(data, lt_cap_text = "A long table (example 3)")
```

You will see in the TeX code that a caption is included in the longtable environment.

4.4.2 Method 2: as a TeX macro

Rather than passing the text for the caption, you can name a macro that should (will) be defined at the time the table is rendered in the TeX document.

Use the `lt_cap_macro` argument:

```
out <- stable_long(data, lt_cap_macro = "ltexfourcap")
```

Now, there is a call to that macro in the table and you **must** define that macro prior to sourcing in your tex document.

```
\newcommand{\ltexfourcap}{
  Another long table - example 4
}

\input{example4.tex}
```

4.5 Add a label

To add a label, pass in a caption (either as text or as a macro) and then use the `lt_cap_label` argument:

```
data %>%
  stable_long(lt_cap_label = "tab:example", lt_cap_macro = "example")
```

4.6 Include a longtable in an Rmd document

If you want to dump a table into an R markdown document, pass it to `st_asis()`. There is a method for `stable_long()` and it will wrap the table properly.

```
out <- stable_long(data) %>% st_asis()
```

4.7 Page breaks

Starting with version 0.4.1, pmtables will modify longtables so that panel headers are kept on the same page as the first row of data underneath the panel row.

5 The pipe interface

5.1 Basics

Mostly working with this data; but some others come in later to illustrate certain features.

```
data <- pmt_summarized
head(data)
```

```
. # A tibble: 6 x 9
.   STUDY       DOSE   FORM    N     WT    CRCL  AGE   ALB   SCR
.   <chr>       <chr>  <chr>   <chr> <chr> <chr> <chr> <chr> <chr>
. 1 12-DEMO-001 100 mg tablet  80    71.4  104   33.7  4.20  1.06
. 2 12-DEMO-001 150 mg capsule 16    89.4  122   24.4  4.63  1.12
. 3 12-DEMO-001 150 mg tablet  48    81.7  104   34.4  3.83  0.910
. 4 12-DEMO-001 150 mg troche  16    94.0  93.2  27.4  4.94  1.25
. 5 12-DEMO-001 200 mg tablet  64    67.9  100   27.5  4.25  1.10
. 6 12-DEMO-001 200 mg troche  16    76.6  99.2  22.8  4.54  1.15
```

You start out a pipeline by passing your data frame into `st_new()`

```
data %>% st_new() %>% class
```

```
. [1] "stobject"    "environment"
```

This creates an object that gets revised by subsequent steps in the pipeline, adding features and styling as you go.

For the final step in the pipeline, we'll send the object to `stable()` to create the table

```
data %>% st_new() %>% stable() %>% head(n=9)
```

```
. [1] "\\setlength{\\tabcolsep}{5pt} "
. [2] "\\begin{threeparttable}"
. [3] "\\renewcommand{\\arraystretch}{1.3}"
. [4] "\\begin{tabular}[h]{lllllllll}"
. [5] "\\hline"
. [6] "STUDY & DOSE & FORM & N & WT & CRCL & AGE & ALB & SCR \\\\"
. [7] "\\hline"
. [8] "12-DEMO-001 & 100 mg & tablet & 80 & 71.4 & 104 & 33.7 & 4.20 & 1.06 \\\\"
. [9] "12-DEMO-001 & 150 mg & capsule & 16 & 89.4 & 122 & 24.4 & 4.63 & 1.12 \\\\"
```

5.2 Simple table

You can terminate the pipeline and create a tabular table by sending to `stable()`

```
data %>% st_new() %>% stable()
```

Equivalent call

```
data %>% stable()
```

5.3 Long table

You can also pipe to `stable_long()` to make a long table

```
data %>% st_new() %>% stable_long()
```

5.4 Annotate with file names

```
st_new(data) %>% st_files(r = "foo.R", output = "foo.tex")
```

Equivalent call

```
stable(data, r_file = "foo.R", output_file = "foo.tex")
```

Note that in the pipe version, we already have `file` in the function name so that gets dropped from the argument name.

5.5 Save a table

First convert with `stable()` or `stable_long()` then save with `stable_save()`

```
st_new(data) %>%
  st_files(output = "foo.tex") %>%
  stable() %>%
  stable_save()
```

5.6 Align columns

- `st_center(...)`
- `st_left(...)`
- `st_right(...)`
- `st_align(...)`

```
st_new(data) %>%
  st_center() %>%
  stable()
```

Equivalent call

```
stable(data, align = cols_center())
```

5.7 Rename columns

```
st_new(data) %>%
  st_rename(Weight = "WT") %>%
  stable()
```

Equivalent call

```
stable(data, cols_rename = c(Weight = "WT"))
```

5.8 Blank columns

```
st_new(data) %>% st_blank(WT, ALB, DOSE) %>% stable()
```

Equivalent call

```
stable(data, cols_blank = "WT,ALB,DOSE")
```

5.9 Drop columns

```
st_new(data) %>% st_drop(WT, ALB, DOSE) %>% stable()
```

Equivalent call

```
stable(data, drop = "WT, ALB, DOSE")
```

5.10 Notes

- Multiple calls are allowed; notes will accumulate

```
st_new(data) %>%
  st_notes("first note") %>%
  st_notes("second note") %>%
  stable()
```

Equivalent call

```
stable(
  data,
  notes = c("first note", "second note")
)
```

5.11 Units

```
st_new(data) %>%
  st_units(WT = "kg", AGE = "years") %>%
  stable()
```

Note that `st_units()` will automatically add parens to your units; this can be suppressed with the `parens` argument.

Units can also be added as a list

```
u <- list(WT = "kg", AGE = "years")

st_new(data) %>%
  st_units(u) %>%
  stable()
```

Equivalent call

```
stable(data, units = u)
```

5.12 panel

```
st_new(data) %>%
  st_panel("STUDY") %>%
  stable()
```

Equivalent call

```
stable(data, panel = "STUDY")
```

5.13 span

```
st_new(data) %>%
  st_span("Covariates", c(WT, ALB, EGFR)) %>%
  stable()
```

Equivalent call

```
stable(data, span = as.span("Covariates", c(WT, ALB, EGFR)))
```

5.14 hlines - at

To put horizontal lines at specific rows

```
st_new(data) %>% st_hline(at = c(2,4,6))
```

Equivalent call

```
stable(data, hline_at = c(2,4,6))
```

5.15 hlines - from

To calculate hlines based on data frame column values (for example, to break the table by changing values of `STUDYf`)

```
st_new(data) %>% st_hline(from = "STUDYf")
```

Equivalent call

```
stable(data, hline_from = "STUDYf")
```

5.16 hlines - pattern

To search the table for a pattern and add hlines above matches

```
st_new(data) %>% st_hline(pattern  = "All", cols = "Summary")
```

The `cols` argument limits the search to the `Summary` column; omit this argument to search the whole table.

There is no equivalent call for this using straight `stable()`.

5.17 Clear replicate values

To create groups by "clearing" replicate values in a column

```
st_new(data) %>% st_clear_reps(STUDY) %>% stable()
```

Equivalent call

```
stable(data, clear_reps = "STUDY")
```

6 Pipe cheat table

Call	Arguments	Description
st_new()	data ...	Create an ‘st‘ object.
st_data()	data ...	Create an ‘st‘ object.
st_files()	r output esc	Add file name information to an ‘st‘ object.
st_sizes()	...	Resize various table elements.
st_space()	row col	Add row and column spacing information to ‘st‘ object.
st_panel()	...	Add panel (row group) information to ‘st‘ object.
st_hline()	pattern cols n at from nudge newline Use pattern to search a column for the location. newline Use at to indicate row numbers or a logical vector indicating rows.	Add horizontal lines to the table.
st_span()	... split	Add spanner (column group) information to ‘st‘ object.
st_span_split()	... split	Add spanner data by splitting column names.
st_sumrow()	pattern cols rows ...	Identify a summary row for the table.
st_rename()	... .list	Rename table columns.
st_drop()	...	Drop entire columns from the table.
st_blank()	...	Clear specific column names from the table header.
st_units()	... parens	Add unit information to table header.
st_clear_reps()	... .now	Clear replicate data within a column.

continued on next page

Call	Arguments	Description
st_left()	...	Make columns left aligned.
st_right()	...	Make columns right aligned.
st_center()	...	Make columns centered.
st_align()	...	Pass general alignment information.
st_notes()	... esc config collapse	Add note information to an ‘st‘ object.
st_noteconf()	...	Add note configuration information to an ‘st‘ object.
st_select()	...	Select columns in an ‘st‘ object.
st_mutate()	...	Mutate columns in an ‘st‘ object.
st_edit()	...	Edit table contents.
st_bold()	cols pattern	Render table data in bold font.
st_it()	cols pattern	Render table data in italic font.
st_args()	...	Pass other arguments to ‘stable()‘.
stable()	data ...	Render table in tabular environment.
stable_long()	data ...	Render table in longtable environment.

7 Options for previewing the table

7.1 st2report

Take a table or a list of tables and render them in a report-like document with a table of contents and a (faux) caption for each table

```
data %>% stable() %>% st2report()
```

You might try passing `ntex` to force the document to build more than once (sometimes the layout settles down after the second build

```
data %>% stable() %>% st2report(ntex = 2)
```

Pass in a list of tables and you will get one table on each page with a listing of tables in the table of contents

```
list(table1, table2, table3) %>% st2report()
```

7.2 st2viewer

This function relies on `texPreview()` to render your table and display it as a graphic in the viewer window

```
data %>% stable() %>% st2viewer()
```

This method is more convenient because the tables always go to the viewer. But the rendering will not be like what you will see in the report.

7.3 st2article

Like `st2report()` but less report-like. You should use `st2report()` instead.

```
data %>% stable() %>% st2article()
```

7.4 st2doc

The original. Rather than building a TeX article, it runs the table in a `Rmd` document via pandoc. Not recommended; it is much slower to get the preview because there has to be a call to `pandoc`

```
data %>% stable() %>% st2doc()
```

8 A word about sanitizing table contents

8.1 Notes

```
x <- ptdata() %>% st_new(notes = "EDA_summary = TRUE") %>%
  st_make(inspect = TRUE) %>%
  get_stable_data()

x$notes
```

```
. [1] "EDA\\_summary = TRUE"
```

8.2 File names

```
x <- ptdata() %>% st_new() %>%
  st_files(r = "my_script.R") %>%
  st_make(inspect = TRUE) %>%
  get_stable_data()

x$notes
```

```
. [1] "Source code: my\\_script.R"
```

8.3 Column names

```
out <-
  tibble(a_1 = 5) %>%
  stable(inspect = TRUE) %>%
  get_stable_data()

out$cols_tex
```

```
. [1] "a\\_1 \\\\"
```

8.4 Main table contents

```
out <-
  tibble(a = "5_2") %>%
  stable(inspect = TRUE) %>%
  get_stable_data()

out$tab
```

```
. [1] "5\\_2 \\\\"
```

8.5 Span titles

```
out <-
  ptdata() %>%
  stable(inspect = TRUE, span = colgroup("foo_this", WT:SCR)) %>%
  get_stable_data()

out$span_data$tex
```

```
. [1] "\\multicolumn{4}{c}{} & \\multicolumn{5}{c}{foo\\_this} \\\\"
. [2] "\\cmidrule(lr){5-9}"
```

8.6 Panel names

```
data <- tibble(a = c("a_1", "a_1", "a_1", "a_2", "a_2"),
               b = letters[1:5])

out <- stable(data, panel = "a")
out[grepl("multicolumn", out)]
```

```
. [1] "\\multicolumn{1}{l}{\\textbf{a\\_1}}\\\\%--pmtables-insert-panel"
. [2] "\\hline \\multicolumn{1}{l}{\\textbf{a\\_2}}\\\\%--pmtables-insert-panel"
```

8.7 `cols_extra` input

```
x <- letters[1:5]
data <- tibble(a = x, b = x, c = x)
xtra <- tibble(a = "foo%", b = "$\\mu$g", c = "1234 \\% %")
out <- stable(data, cols_extra = xtra)
out[grepl("%", out, fixed = TRUE)]
```

```
. [1] "foo\\% & $\\mu$g & 1234 \\% % \\\\"
```

9 pmtable

Some setup

```
units = yspec::ys_get_unit(yspec::ys_help$spec(), parens = TRUE)
```

```
data <- pmt_first
data_pk <- pmt_pk
data_all <- pmt_obs
```

9.1 Principles

These functions expect that the user passes in all data that is to be summarized and nothing more. We will not filter your data.

9.2 Rename cols

When you select columns to summarize, you can generally pass in alternate (nicer) names that you want to show up in the table. For example, if I have a column called `WT` in the data frame and I want it to show up as `Weight` this can be accomplished during the call

```
pt_cont_wide(data, cols = c(Weight = "WT")) %>%
  stable(notes = NULL) %>% st_asis()
```

Weight
70.7 (12.8) [157]

Alternatively, you can use the `table` argument to enter rename info. Note that `table` is a list that should have names that match up with columns in the data frame and values that are the new names

```
tab <- list(SEXf = "Sex", ASIANf = "Race group")

pt_cat_wide(data, cols = "SEXf,ASIANf", table = tab) %>%
  stable() %>% st_asis()
```

	Sex		Race group	
n	male	female	Asian	non-Asian
160	80 (50.0)	80 (50.0)	66 (41.2)	94 (58.8)

Summary is count (percent)
n: number of records summarized

9.3 Data inventory tables

- Count number of
  - individuals

  - observations
  - BQL observations
  - missing values
- Calculate the percent of observations or BQL in different sub groups

9.3.1 Stacked by endpoint

- The stacked plot creates multiple independent tables to summarize different endpoints; there is no single overall summary for the table because we are summarizing different endpoints

```
out <- pt_data_inventory(
  data_all,
  by = c(Study = "STUDYf"),
  panel = as.panel("SEQf", prefix = "Endpoint: "),
  stacked = TRUE
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Number				Percent	
Study	SUBJ	MISS	OBS	BQL	OBS	BQL
Endpoint: DEMO PK						
12-DEMO-001	30	8	427	15	13.4	0.5
12-DEMO-002	50	10	1152	38	36.3	1.2
11-DEMO-005	40	10	920	30	29.0	0.9
13-DEMO-001	40	7	582	11	18.3	0.3
Group Total	160	35	3081	94	97.0	3.0
Endpoint: ESTRDIOL						
11-DEMO-005	40	0	40	0	50.6	0.0
13-DEMO-001	40	1	39	0	49.4	0.0
Group Total	80	1	79	0	100.0	0.0
Endpoint: BMD						
11-DEMO-005	40	9	111	0	49.1	0.0
13-DEMO-001	40	5	115	0	50.9	0.0
Group Total	80	14	226	0	100.0	0.0

SUBJ: subjects
BQL: below quantification limit
MISS: missing observations (non-BQL)
OBS: observations
Source code: test.R
Source file: test.tex

9.3.2 Paneled

- Just summarize a single endpoint

```
out <- pt_data_inventory(
  data_pk,
  by = c(Study = "STUDYf"),
  panel = "ASIANf"
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Number				Group percent		Overall percent	
Study	SUBJ	MISS	OBS	BQL	OBS	BQL	OBS	BQL
Asian								
12-DEMO-001	17	4	241	10	18.8	0.8	7.6	0.3
12-DEMO-002	18	4	414	14	32.3	1.1	13.0	0.4
11-DEMO-005	16	5	366	13	28.6	1.0	11.5	0.4
13-DEMO-001	15	3	218	4	17.0	0.3	6.9	0.1
non-Asian								
12-DEMO-001	13	4	186	5	9.8	0.3	5.9	0.2
12-DEMO-002	32	6	738	24	38.9	1.3	23.2	0.8
11-DEMO-005	24	5	554	17	29.2	0.9	17.4	0.5
13-DEMO-001	25	4	364	7	19.2	0.4	11.5	0.2
All data	160	35	3081	94	—	—	97.0	3.0

SUBJ: subjects
BQL: below quantification limit
MISS: missing observations (non-BQL)
OBS: observations
Source code: test.R
Source file: test.tex

9.3.3 Grouped (by study)

```
out <- pt_data_inventory(
  data_pk,
  by = c(Study = "STUDYf")
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Number				Percent	
Study	SUBJ	MISS	OBS	BQL	OBS	BQL
12-DEMO-001	30	8	427	15	13.4	0.5
12-DEMO-002	50	10	1152	38	36.3	1.2
11-DEMO-005	40	10	920	30	29.0	0.9
13-DEMO-001	40	7	582	11	18.3	0.3
All data	160	35	3081	94	97.0	3.0

SUBJ: subjects
BQL: below quantification limit
MISS: missing observations (non-BQL)
OBS: observations
Source code: test.R
Source file: test.tex

9.3.4 BQL / BLQ

Beginning with version 0.4.1, pmtables can accommodate either `BQL` or `BLQ` as the name of the column indicating that observations were below the limit of quantitation. Table notes and output column headers will be adjusted based on the input.

For example

```
data_ql <- pmt_obs
data_lq <- dplyr::rename(pmt_obs, BLQ = BQL)
```

```
pt_data_inventory(data_ql, by = "STUDYf") %>%
  st_asis()
```

	Number				Percent	
STUDYf	SUBJ	MISS	OBS	BQL	OBS	BQL
12-DEMO-001	30	8	427	15	12.3	0.4
12-DEMO-002	50	10	1152	38	33.1	1.1
11-DEMO-005	40	19	1071	30	30.8	0.9
13-DEMO-001	40	13	736	11	21.1	0.3
All data	160	50	3386	94	97.3	2.7

SUBJ: subjects
BQL: below quantification limit
MISS: missing observations (non-BQL)
OBS: observations

```
pt_data_inventory(data_lq, by = "STUDYf") %>%
  st_asis()
```

	Number				Percent	
STUDYf	SUBJ	MISS	OBS	BLQ	OBS	BLQ
12-DEMO-001	30	8	427	15	12.3	0.4
12-DEMO-002	50	10	1152	38	33.1	1.1
11-DEMO-005	40	19	1071	30	30.8	0.9
13-DEMO-001	40	13	736	11	21.1	0.3
All data	160	50	3386	94	97.3	2.7

SUBJ: subjects
BLQ: below limit of quantification
MISS: missing observations (non-BLQ)
OBS: observations

9.4 Wide categorical table

- Summary of categorical data in wide format
- The summary is `number (percent within group)`
- Wide refers to the fact that the covariates go across the table

9.4.0.1 Ungrouped

```
out <- pt_cat_wide(
  data = data,
  cols = vars(Formulation = FORMf, Sex = SEXf, "Race group" = ASIANf)
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Formulation			Sex		Race group	
n	tablet	capsule	troche	male	female	Asian	non-Asian
160	130 (81.2)	15 (9.4)	15 (9.4)	80 (50.0)	80 (50.0)	66 (41.2)	94 (58.8)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.4.1 Paneled (limited utility, IMO)

- Provided here for completeness

```
out <- pt_cat_wide(
  data = data,
  cols = vars(Formulation = FORMf, Sex = SEXf, "Race group" = ASIANf),
  panel = as.panel("STUDYf", prefix = "Study: ")
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Formulation			Sex		Race group	
n	tablet	capsule	troche	male	female	Asian	non-Asian
Study: 12-DEMO-001							
30	25 (83.3)	3 (10.0)	2 (6.7)	10 (33.3)	20 (66.7)	17 (56.7)	13 (43.3)
Study: 12-DEMO-002							
50	42 (84.0)	6 (12.0)	2 (4.0)	18 (36.0)	32 (64.0)	18 (36.0)	32 (64.0)
Study: 11-DEMO-005							
40	30 (75.0)	3 (7.5)	7 (17.5)	29 (72.5)	11 (27.5)	16 (40.0)	24 (60.0)
Study: 13-DEMO-001							
40	33 (82.5)	3 (7.5)	4 (10.0)	23 (57.5)	17 (42.5)	15 (37.5)	25 (62.5)
All data							
160	130 (81.2)	15 (9.4)	15 (9.4)	80 (50.0)	80 (50.0)	66 (41.2)	94 (58.8)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.4.2 Grouped (by male / female)

```
out <- pt_cat_wide(
  data = data,
  by = c(Sex = "SEXf"),
  cols = vars(Formulation = FORMf, "Race group" = ASIANf)
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

		Formulation			Race group	
Sex	n	tablet	capsule	troche	Asian	non-Asian
male	80	62 (77.5)	7 (8.8)	11 (13.8)	28 (35.0)	52 (65.0)
female	80	68 (85.0)	8 (10.0)	4 (5.0)	38 (47.5)	42 (52.5)
All data	160	130 (81.2)	15 (9.4)	15 (9.4)	66 (41.2)	94 (58.8)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.4.3 Paneled and grouped

```
out <- pt_cat_wide(
  data = data,
  cols = vars(Formulation = FORMf, Sex = SEXf,"Race group" = ASIANf),
  panel = as.panel("STUDYf", prefix = "Study: "),
  by = c("RF Group" = "RFf")
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

		Formulation			Sex		Race group	
RF Group	n	tablet	capsule	troche	male	female	Asian	non-Asian
Study: 12-DEMO-001								
normal	30	25 (83.3)	3 (10.0)	2 (6.7)	10 (33.3)	20 (66.7)	17 (56.7)	13 (43.3)
Study: 12-DEMO-002								
normal	50	42 (84.0)	6 (12.0)	2 (4.0)	18 (36.0)	32 (64.0)	18 (36.0)	32 (64.0)
Study: 11-DEMO-005								
normal	10	9 (90.0)	0 (0.0)	1 (10.0)	7 (70.0)	3 (30.0)	3 (30.0)	7 (70.0)
mild	10	7 (70.0)	2 (20.0)	1 (10.0)	7 (70.0)	3 (30.0)	5 (50.0)	5 (50.0)
moderate	10	6 (60.0)	0 (0.0)	4 (40.0)	8 (80.0)	2 (20.0)	6 (60.0)	4 (40.0)
severe	10	8 (80.0)	1 (10.0)	1 (10.0)	7 (70.0)	3 (30.0)	2 (20.0)	8 (80.0)
Study: 13-DEMO-001								
normal	40	33 (82.5)	3 (7.5)	4 (10.0)	23 (57.5)	17 (42.5)	15 (37.5)	25 (62.5)
All data	160	130 (81.2)	15 (9.4)	15 (9.4)	80 (50.0)	80 (50.0)	66 (41.2)	94 (58.8)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.4.4 No summary

```
out <- pt_cat_wide(
  data = data,
  summarize = "none",
  cols = vars(Formulation = FORMf, Sex = SEXf,"Race group" = ASIANf),
  panel = as.panel("STUDYf", prefix = "Study: "),
  by = c("RF Group" = "RFf")
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

		Formulation			Sex		Race group	
RF Group	n	tablet	capsule	troche	male	female	Asian	non-Asian
Study: 12-DEMO-001								
normal	30	25 (83.3)	3 (10.0)	2 (6.7)	10 (33.3)	20 (66.7)	17 (56.7)	13 (43.3)
Study: 12-DEMO-002								
normal	50	42 (84.0)	6 (12.0)	2 (4.0)	18 (36.0)	32 (64.0)	18 (36.0)	32 (64.0)
Study: 11-DEMO-005								
normal	10	9 (90.0)	0 (0.0)	1 (10.0)	7 (70.0)	3 (30.0)	3 (30.0)	7 (70.0)
mild	10	7 (70.0)	2 (20.0)	1 (10.0)	7 (70.0)	3 (30.0)	5 (50.0)	5 (50.0)
moderate	10	6 (60.0)	0 (0.0)	4 (40.0)	8 (80.0)	2 (20.0)	6 (60.0)	4 (40.0)
severe	10	8 (80.0)	1 (10.0)	1 (10.0)	7 (70.0)	3 (30.0)	2 (20.0)	8 (80.0)
Study: 13-DEMO-001								
normal	40	33 (82.5)	3 (7.5)	4 (10.0)	23 (57.5)	17 (42.5)	15 (37.5)	25 (62.5)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.5 Long categorical table

- Categorical table in long format
- Long indicates that the covariates go down the table

9.5.1 Ungrouped

```
out <- pt_cat_long(
  data = data,
  cols = vars(Study = STUDYf, Sex = SEXf, "Race group" = ASIANf, "Child-Pugh" = CPf)
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Summary n = 160
Study	
12-DEMO-001	30 (18.8)
12-DEMO-002	50 (31.2)
11-DEMO-005	40 (25.0)
13-DEMO-001	40 (25.0)
Sex	
male	80 (50.0)
female	80 (50.0)
Race group	
Asian	66 (41.2)
non-Asian	94 (58.8)
Child-Pugh	
Score=0	130 (81.2)
Score=1	10 (6.2)
Score=2	10 (6.2)
Score=3	10 (6.2)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.5.2 Grouped (by formulation)

```
out <- pt_cat_long(
  data = data,
  cols = vars(Study = STUDYf,Sex = SEXf,"Race group" = ASIANf, "Child-Pugh" = CPf),
  span = c(Formulation = "FORMf")
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Formulation			
	tablet n = 130	capsule n = 15	troche n = 15	Summary n = 160
Study				
12-DEMO-001	25 (19.2)	3 (20.0)	2 (13.3)	30 (18.8)
12-DEMO-002	42 (32.3)	6 (40.0)	2 (13.3)	50 (31.2)
11-DEMO-005	30 (23.1)	3 (20.0)	7 (46.7)	40 (25.0)
13-DEMO-001	33 (25.4)	3 (20.0)	4 (26.7)	40 (25.0)
Sex				
male	62 (47.7)	7 (46.7)	11 (73.3)	80 (50.0)
female	68 (52.3)	8 (53.3)	4 (26.7)	80 (50.0)
Race group				
Asian	53 (40.8)	7 (46.7)	6 (40.0)	66 (41.2)
non-Asian	77 (59.2)	8 (53.3)	9 (60.0)	94 (58.8)
Child-Pugh				
Score=0	106 (81.5)	12 (80.0)	12 (80.0)	130 (81.2)
Score=1	7 (5.4)	1 (6.7)	2 (13.3)	10 (6.2)
Score=2	8 (6.2)	1 (6.7)	1 (6.7)	10 (6.2)
Score=3	9 (6.9)	1 (6.7)	0 (0.0)	10 (6.2)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.5.3 Summary on bottom and right

```
out <- pt_cat_long(
  data = data,
  summarize = "both",
  cols = vars(Formulation = FORMf, Sex = SEXf,"Race group" = ASIANf),
  span = vars(Study = STUDYf)
  )

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Study				
	12-DEMO-001 n = 30	12-DEMO-002 n = 50	11-DEMO-005 n = 40	13-DEMO-001 n = 40	Summary n = 160
Formulation					
tablet	25 (83.3)	42 (84.0)	30 (75.0)	33 (82.5)	130 (81.2)
capsule	3 (10.0)	6 (12.0)	3 (7.5)	3 (7.5)	15 (9.4)
troche	2 (6.7)	2 (4.0)	7 (17.5)	4 (10.0)	15 (9.4)
Sex					
male	10 (33.3)	18 (36.0)	29 (72.5)	23 (57.5)	80 (50.0)
female	20 (66.7)	32 (64.0)	11 (27.5)	17 (42.5)	80 (50.0)
Race group					
Asian	17 (56.7)	18 (36.0)	16 (40.0)	15 (37.5)	66 (41.2)
non-Asian	13 (43.3)	32 (64.0)	24 (60.0)	25 (62.5)	94 (58.8)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.5.4 No summary

```
out <- pt_cat_long(
  data = data,
  summarize = "none",
  cols = vars(Formulation = FORMf, Sex = SEXf,"Race group" = ASIANf),
  span = vars(Study = STUDYf)
  )

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

	Study			
	12-DEMO-001	12-DEMO-002	11-DEMO-005	13-DEMO-001
Formulation				
tablet	25 (83.3)	42 (84.0)	30 (75.0)	33 (82.5)
capsule	3 (10.0)	6 (12.0)	3 (7.5)	3 (7.5)
troche	2 (6.7)	2 (4.0)	7 (17.5)	4 (10.0)
Sex				
male	10 (33.3)	18 (36.0)	29 (72.5)	23 (57.5)
female	20 (66.7)	32 (64.0)	11 (27.5)	17 (42.5)
Race group				
Asian	17 (56.7)	18 (36.0)	16 (40.0)	15 (37.5)
non-Asian	13 (43.3)	32 (64.0)	24 (60.0)	25 (62.5)

Summary is count (percent)
n: number of records summarized
Source code: test.R
Source file: test.tex

9.6 Wide continuous table

- Continuous table in wide format
- Wide means that the covariates go across the table

9.6.1 Ungrouped

```
out <- pt_cont_wide(
  data = data,
  cols = "WT,SCR,AGE,ALB,HT",
  units = units
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

WT (kg)	SCR (mg/dL)	AGE (years)	ALB (g/dL)	HT (cm)
70.7 (12.8) [157]	1.36 (0.986) [160]	33.7 (8.83) [160]	4.20 (0.793) [156]	179 (17.7) [160]

Summary is mean (sd) [count]
Source code: test.R
Source file: test.tex

9.6.2 Paneled

```
out <- pt_cont_wide(
  data = data,
  cols = "WT,SCR,AGE,ALB,HT",
  panel = c(Study = "STUDYf"),
  units = units
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

WT (kg)	SCR (mg/dL)	AGE (years)	ALB (g/dL)	HT (cm)
Study 12-DEMO-001				
72.2 (14.3) [29]	1.03 (0.155) [30]	32.0 (9.19) [30]	4.28 (0.474) [29]	180 (19.3) [30]
Study 12-DEMO-002				
72.4 (11.5) [49]	0.971 (0.161) [50]	35.0 (8.20) [50]	4.47 (0.468) [50]	182 (15.4) [50]
Study 11-DEMO-005				
68.9 (14.5) [39]	2.52 (1.43) [40]	32.8 (8.48) [40]	4.41 (0.537) [39]	175 (19.2) [40]
Study 13-DEMO-001				
69.4 (11.6) [40]	0.950 (0.165) [40]	34.2 (9.67) [40]	3.58 (1.15) [38]	179 (17.2) [40]
All data				
70.7 (12.8) [157]	1.36 (0.986) [160]	33.7 (8.83) [160]	4.20 (0.793) [156]	179 (17.7) [160]

Summary is mean (sd) [count]
Source code: test.R
Source file: test.tex

9.6.3 Grouped (by study)

```
out <- pt_cont_wide(
  data = data,
  cols = "WT,SCR,AGE,ALB,HT",
  by = c(Study = "STUDYf"),
  units = units
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

Study	WT (kg)	SCR (mg/dL)	AGE (years)	ALB (g/dL)	HT (cm)
12-DEMO-001	72.2 (14.3) [29]	1.03 (0.155) [30]	32.0 (9.19) [30]	4.28 (0.474) [29]	180 (19.3) [30]
12-DEMO-002	72.4 (11.5) [49]	0.971 (0.161) [50]	35.0 (8.20) [50]	4.47 (0.468) [50]	182 (15.4) [50]
11-DEMO-005	68.9 (14.5) [39]	2.52 (1.43) [40]	32.8 (8.48) [40]	4.41 (0.537) [39]	175 (19.2) [40]
13-DEMO-001	69.4 (11.6) [40]	0.950 (0.165) [40]	34.2 (9.67) [40]	3.58 (1.15) [38]	179 (17.2) [40]
All data	70.7 (12.8) [157]	1.36 (0.986) [160]	33.7 (8.83) [160]	4.20 (0.793) [156]	179 (17.7) [160]

Summary is mean (sd) [count]
Source code: test.R
Source file: test.tex

9.6.4 Paneled and grouped

```
out <- pt_cont_wide(
  data = data,
  cols = "WT,SCR,AGE,ALB,HT",
  by = c(Study = "STUDYf"),
  panel = c(Formulation = "FORMf"),
  units = units
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

Study	WT (kg)	SCR (mg/dL)	AGE (years)	ALB (g/dL)	HT (cm)
Formulation tablet					
12-DEMO-001	71.0 (14.2) [24]	1.01 (0.157) [25]	32.6 (9.23) [25]	4.22 (0.459) [24]	179 (19.7) [25]
12-DEMO-002	72.2 (11.8) [41]	0.966 (0.166) [42]	34.0 (7.93) [42]	4.49 (0.495) [42]	182 (15.9) [42]
11-DEMO-005	68.8 (15.2) [29]	2.48 (1.47) [30]	33.2 (8.73) [30]	4.37 (0.568) [29]	173 (19.7) [30]
13-DEMO-001	69.4 (11.0) [33]	0.967 (0.163) [33]	33.7 (9.67) [33]	3.53 (1.14) [31]	178 (16.5) [33]
Formulation capsule					
12-DEMO-001	72.9 (17.3) [3]	1.12 (0.0700) [3]	32.2 (12.0) [3]	4.49 (0.593) [3]	184 (23.0) [3]
12-DEMO-002	70.9 (10.3) [6]	1.03 (0.146) [6]	37.7 (7.59) [6]	4.38 (0.354) [6]	181 (15.4) [6]
11-DEMO-005	73.9 (11.1) [3]	3.06 (2.19) [3]	31.8 (4.99) [3]	4.65 (0.240) [3]	181 (16.4) [3]
13-DEMO-001	58.4 (4.04) [3]	0.973 (0.195) [3]	36.5 (6.69) [3]	3.09 (1.50) [3]	167 (8.88) [3]
Formulation troche					
12-DEMO-001	85.3 (12.4) [2]	1.20 (0.0707) [2]	25.1 (3.28) [2]	4.74 (0.283) [2]	194 (0.163) [2]
12-DEMO-002	79.7 (8.61) [2]	0.910 (0.0283) [2]	48.0 (1.79) [2]	4.49 (0.0354) [2]	182 (10.9) [2]
11-DEMO-005	66.8 (13.9) [7]	2.45 (1.05) [7]	31.4 (9.34) [7]	4.49 (0.509) [7]	177 (19.8) [7]
13-DEMO-001	77.4 (15.9) [4]	0.795 (0.0777) [4]	37.3 (12.9) [4]	4.32 (0.994) [4]	193 (22.4) [4]
All data	70.7 (12.8) [157]	1.36 (0.986) [160]	33.7 (8.83) [160]	4.20 (0.793) [156]	179 (17.7) [160]

Summary is mean (sd) [count]
Source code: test.R
Source file: test.tex

9.7 Long continuous table

- Continuous summary table in long format
- Long indicates that covariates go down the table

9.7.1 Ungrouped

```
out <- pt_cont_long(
  data = data,
  cols = "WT,SCR,AGE",
  units = units
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

Variable	n	Mean	Median	SD	Min / Max
WT (kg)	157	70.7	70.0	12.8	43.6 / 97.2
SCR (mg/dL)	160	1.36	1.04	0.986	0.710 / 5.59
AGE (years)	160	33.7	33.4	8.83	18.9 / 49.5

n: number of records summarized
SD: standard deviation
Min: minimum; Max: maximum
Source code: test.R
Source file: test.tex

9.7.2 Paneled

```
out <- pt_cont_long(
  data = data,
  cols = "WT,SCR,AGE",
  panel = vars(Study = STUDYf),
  units = units
)

out %>% stable(r_file = "test.R", output_file = "test.tex") %>% st_asis()
```

Variable	n	Mean	Median	SD	Min / Max
Study 12-DEMO-001					
WT (kg)	29	72.2	70.0	14.3	50.9 / 97.2
SCR (mg/dL)	30	1.03	1.04	0.155	0.740 / 1.30
AGE (years)	30	32.0	28.0	9.19	19.9 / 47.8
Study 12-DEMO-002					
WT (kg)	49	72.4	72.1	11.5	51.5 / 96.6
SCR (mg/dL)	50	0.971	0.970	0.161	0.720 / 1.30
AGE (years)	50	35.0	36.0	8.20	20.3 / 49.2
Study 11-DEMO-005					
WT (kg)	39	68.9	65.4	14.5	43.6 / 92.8
SCR (mg/dL)	40	2.52	2.33	1.43	0.720 / 5.59
AGE (years)	40	32.8	33.4	8.48	19.2 / 49.5
Study 13-DEMO-001					
WT (kg)	40	69.4	68.1	11.6	50.7 / 96.6
SCR (mg/dL)	40	0.950	0.975	0.165	0.710 / 1.26
AGE (years)	40	34.2	35.2	9.67	18.9 / 49.5
All data					
WT (kg)	157	70.7	70.0	12.8	43.6 / 97.2
SCR (mg/dL)	160	1.36	1.04	0.986	0.710 / 5.59
AGE (years)	160	33.7	33.4	8.83	18.9 / 49.5

n: number of records summarized
SD: standard deviation
Min: minimum; Max: maximum
Source code: test.R
Source file: test.tex

9.8 Demographics table

A demographics table summarizes both continuous and discrete data in a single table.

- Both continuous columns (`cols_cont`) and discrete (`cols_cat`) are required
- You can specify a `span` column (the table is pretty skinny without that)
- You can opt out of the paneling too with `paneled` argument (the also makes the table wider
- An `All data` summary is provided on the left (opt out with `summarize_all`)

9.8.1 With span

```
pt_demographics(
  pmt_first,
  cols_cont = "WT, CRCL",
  cols_cat = "SEXf, CPf",
  span = c(Study = "STUDYf")
) %>% st_asis()
```

	Study				
Statistic	12-DEMO-001 n = 30	12-DEMO-002 n = 50	11-DEMO-005 n = 40	13-DEMO-001 n = 40	Summary n = 160
WT					
Mean (SD)	72.2 (14.3)	72.4 (11.5)	68.9 (14.5)	69.4 (11.6)	70.7 (12.8)
Min / Max	50.9 / 97.2	51.5 / 96.6	43.6 / 92.8	50.7 / 96.6	43.6 / 97.2
Missing	1	1	1	0	3
CRCL					
Mean (SD)	106 (9.46)	103 (8.35)	58.8 (29.7)	102 (8.19)	92.1 (25.5)
Min / Max	93.2 / 126	90.6 / 121	15.4 / 103	90.7 / 119	15.4 / 126
Missing	1	1	1	3	6
SEXf					
male	10 (33.3)	18 (36.0)	29 (72.5)	23 (57.5)	80 (50.0)
female	20 (66.7)	32 (64.0)	11 (27.5)	17 (42.5)	80 (50.0)
CPf					
Score=0	30 (100.0)	50 (100.0)	40 (100.0)	10 (25.0)	130 (81.2)
Score=1	0 (0.0)	0 (0.0)	0 (0.0)	10 (25.0)	10 (6.2)
Score=2	0 (0.0)	0 (0.0)	0 (0.0)	10 (25.0)	10 (6.2)
Score=3	0 (0.0)	0 (0.0)	0 (0.0)	10 (25.0)	10 (6.2)

Categorical summary is count (percent)
n: number of records summarized
SD: standard deviation
Min: minimum; Max: maximum

9.8.2 No span

- This table is skinny

```
mini <- noteconf(type = "minipage", width = 0.5)
pt_demographics(
  pmt_first,
  cols_cont = "WT, CRCL, AGE",
  cols_cat = "SEXf, CPf"
) %>% stable(note_config = mini) %>% st_asis()
```

Statistic	Summary n = 160
WT	
Mean (SD)	70.7 (12.8)
Min / Max	43.6 / 97.2
Missing	3
CRCL	
Mean (SD)	92.1 (25.5)
Min / Max	15.4 / 126
Missing	6
AGE	
Mean (SD)	33.7 (8.83)
Min / Max	18.9 / 49.5
Missing	0
SEXf	
male	80 (50.0)
female	80 (50.0)
CPf	
Score=0	130 (81.2)
Score=1	10 (6.2)
Score=2	10 (6.2)
Score=3	10 (6.2)

Categorical summary is count (percent)
n: number of records summarized
SD: standard deviation
Min: minimum; Max: maximum

9.8.3 No span, not paneled

- Opting out of the paneling also makes it wider

```
pt_demographics(
  pmt_first,
  cols_cont = "WT, CRCL, AGE",
  cols_cat = "SEXf, CPf",
  paneled = FALSE,
  table = list(WT = "Weight (kg)", CRCL = "CLCR (ml/min)",
               AGE = "Age (years)", SEXf = "Sex", CPf = "Child-Pugh")
) %>% st_asis()
```

Covariate	Statistic	Summary n = 160
Weight (kg)	Mean (SD)	70.7 (12.8)
	Min / Max	43.6 / 97.2
	Missing	3
CLCR (ml/min)	Mean (SD)	92.1 (25.5)
	Min / Max	15.4 / 126
	Missing	6
Age (years)	Mean (SD)	33.7 (8.83)
	Min / Max	18.9 / 49.5
	Missing	0
Sex	male	80 (50.0)
	female	80 (50.0)
Child-Pugh	Score=0	130 (81.2)
	Score=1	10 (6.2)
	Score=2	10 (6.2)
	Score=3	10 (6.2)

Categorical summary is count (percent)
n: number of records summarized
SD: standard deviation
Min: minimum; Max: maximum

9.8.4 No summary

```
pt_demographics(
  pmt_first,
  cols_cont = "WT, CRCL, AGE",
  cols_cat = "SEXf, CPf",
  span = c(Study = "STUDYf"),
  summarize_all = FALSE
) %>% st_asis()
```

	Study			
Statistic	12-DEMO-001 n = 30	12-DEMO-002 n = 50	11-DEMO-005 n = 40	13-DEMO-001 n = 40
WT				
Mean (SD)	72.2 (14.3)	72.4 (11.5)	68.9 (14.5)	69.4 (11.6)
Min / Max	50.9 / 97.2	51.5 / 96.6	43.6 / 92.8	50.7 / 96.6
Missing	1	1	1	0
CRCL				
Mean (SD)	106 (9.46)	103 (8.35)	58.8 (29.7)	102 (8.19)
Min / Max	93.2 / 126	90.6 / 121	15.4 / 103	90.7 / 119
Missing	1	1	1	3
AGE				
Mean (SD)	32.0 (9.19)	35.0 (8.20)	32.8 (8.48)	34.2 (9.67)
Min / Max	19.9 / 47.8	20.3 / 49.2	19.2 / 49.5	18.9 / 49.5
Missing	0	0	0	0
SEXf				
male	10 (33.3)	18 (36.0)	29 (72.5)	23 (57.5)
female	20 (66.7)	32 (64.0)	11 (27.5)	17 (42.5)
CPf				
Score=0	30 (100.0)	50 (100.0)	40 (100.0)	10 (25.0)
Score=1	0 (0.0)	0 (0.0)	0 (0.0)	10 (25.0)
Score=2	0 (0.0)	0 (0.0)	0 (0.0)	10 (25.0)
Score=3	0 (0.0)	0 (0.0)	0 (0.0)	10 (25.0)

Categorical summary is count (percent)
n: number of records summarized
SD: standard deviation
Min: minimum; Max: maximum

9.9 Customized summary functions

pmtables will summarize continuous data using a built-in function, producing standard summaries (e.g. mean, median, etc). Users can pass a function to replace this default, allowing totally customized summaries.

Custom summary functions are not currently allowed for categorical data.

9.9.1 Continuous long table

You can pass a custom summary function via `fun`. This function should have a first argument called `value` and should be able to absorb extra arguments via `...`. The function should return a `data.frame`, with a single row and summaries going across in the columns.

For example, we can have `pt_cont_long()` return the geometric mean and variance by passing the following function

```
cont_long_custom <- function(value, ...) {
  value <- na.omit(value)
  ans <- data.frame(
    GeoMean = exp(mean(log(value))),
    Variance = var(value)
  )
  mutate(ans, across(everything(), sig))
}
```

Test the function by passing some test data

```
cont_long_custom(c(1,2,3,4,5))
```

```
.   GeoMean Variance
. 1    2.61     2.50
```

Then, pass this as `fun`

```
pt_cont_long(
  data = pmt_first,
  cols = c("WT", "ALB", "AGE"),
  fun = cont_long_custom
)$data
```

```
. # A tibble: 3 x 3
.   Variable GeoMean Variance
.   <chr>    <chr>   <chr>
. 1 WT       69.5    165
. 2 ALB      4.11    0.629
. 3 AGE      32.5    78.0
```

See `pmtables:::cont_long_fun` (the default) for an example.

9.9.2 Continuous wide table

You can pass a custom summary function via `fun`. This function should have a first argument called `value` and should be able to absorb extra arguments via `...`. The continuous, wide table must return a `data.frame` with a single row and a single column named `summary`

```
cont_wide_custom <- function(value, ...) {
  value <- na.omit(value)
  geo_mean <- sig(exp(mean(log(value))))
  variance <- sig(var(value))
  n <- length(value)
  ans <- paste0(geo_mean, " [", variance, "] (", n, ")")
  data.frame(summary = ans)
}
```

You can test the function by passing some test data

```
cont_wide_custom(c(1, 3, 5))
```

```
.           summary
. 1 2.47 [4.00] (3)
```

Then, pass this as `fun`

```
pt_cont_wide(
  data = pmt_first,
  cols = c("WT", "ALB", "AGE"),
  fun = cont_wide_custom
)$data
```

```
. # A tibble: 1 x 3
.   WT                ALB                 AGE
.   <chr>             <chr>               <chr>
. 1 69.5 [165] (157) 4.11 [0.629] (156) 32.5 [78.0] (160)
```

See `pmtables:::cont_wide_fun` (the default) for an example.