The pmtables Book

Author: Kyle Baron

Date: 2021-03-06

Version: 0.3.2.9100

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 thos 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 untill 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 colum 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 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

1.6.4 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

1.6.5 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 appeaance 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

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: 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 arument 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("The Universe", Tariffville:Minneapolis, level = 2)
)
data %>% stable(span = sp) %>% st_asis()
```

	The Universe		
	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("The Universe", Tariffville:Minneapolis, level = 2) %>%
  stable() %>%
  st_asis()
```

The Universe			
	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("Example ... States", Connecticut:Minnesota)) %>%
  st_asis()
```

Example States			
Tariffville	Connecticut	Minnesota	Minneapolis
06081	CT	MN	55455

4 Tables that span multiple pages: longtable

4.1 Syntax

To create a long table

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

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}
```

You should not wrap your longtable input code in `\begin{table}`/`\end{table}` as you would with a regular tabular table.

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 pipline 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_nates("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)))
```

6 Options for previewing the table

6.1 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()
```

6.2 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 conveient because the tables always go to the viewer. But the rendering will not be like what you will see in the report.

6.3 st2article

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

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

6.4 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()
```

7 A word about sanitizing table contents

7.1 Notes

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

x$notes
```

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

7.2 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"
```

7.3 Column names

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

out$cols_tex
```

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

7.4 Main table contents

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

out$tab
```

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

7.5 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}"
```

7.6 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}}\\\\"
. [2] "\\hline \\multicolumn{1}{l}{\\textbf{a\\_2}}\\\\"
```

7.7 `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 \\% % \\\\"
```