The **htmlTable** 2.0 package was just released on CRAN! It is my most downloaded package with 160 000+ downloads/month and this update is something that I have been wanting to do for a long time. For those of you that never encountered **htmlTable** it is a package that takes a `matrix`/`data.frame` and outputs a nicely formatted HTML table. When I created the package there weren't that many alternatives and **knitr** was this new thing that everyone was excited about, **magrittr** with its ubiquitous `%>%` pipe had not even entered the scene. The current update should make it easier to streamline table look, separate layout from content and use **tidyverse** functionality.

Style and theming

The biggest change in how to use the `htmlTable` is that you have a separate function for adding style to the table. The change is implemented in a non-breaking fashion, i.e. all the old options should work just as before but the new API is encouraged as it simplifies a lot. The new API changes so that instead of all the `css.*` arguments we now have a separate function that applies these, `addHtmlTableStyle` and passes them on as an attribute to the `htmlTable`:

```
library(htmlTable)
library(magrittr)
options(table_counter = TRUE)

rbind(
  `Group A` = c(20, 5, 380, 95),
  `1` = c(11, 55, 9, 45),
  `2` = c(11, 55, 9, 45)
) %>%
  addHtmlTableStyle(css.rgroup = "font-style: italic",
                    css.header = "font-weight: normal") %>%
  htmlTable(header = rep(c("No", "%"), times = 2),
            n.cgroup = list(c(2), c(2, 2)),
            cgroup = list('Super', c("First", "Second")),
            rgroup = c("", "Group B"),
            n.rgroup = 1,
            caption = "A simple htmlTable example with the core components")
```

Table 1: A simple htmlTable example with the core components

	Super			
	First		Second	
	No	%	No	%
Group A	20	5	380	95
Group B				
1	11	55	9	45
2	11	55	9	45

As we usually want the same layout for the entire table we can accomplish the same effect for all of our tables using `setHtmlTableTheme` and the style will be applied to all your tables.

```
library(glue)
setHtmlTableTheme(css.rgroup = "font-style: italic",
                  css.header = "font-weight: normal",
                  pos.caption = "bottom")


rbind(
  `Group A` = c(20, 5, 380, 95),
  `1` = c(11, 55, 9, 45),
  `2` = c(11, 55, 9, 45)
```

```
) %>%
  htmlTable(header = rep(c("No", "%"), times = 2),
            n.cgroup = list(c(2), c(2, 2)),
            cgroup = list('Super', c("First", "Second")),
            rgroup = c("", "Group B"),
            n.rgroup = 1,
            caption = glue("Same as Table {last} but with general styling and caption
positioned at the bottom.",
                           last = tblNoLast()))
```

	Super			
	First		Second	
	No	%	No	%
Group A	20	5	380	95
Group B				
1	11	55	9	45
2	11	55	9	45

Table 2: Same as Table 1 but with general styling and caption positioned at the bottom.

There is an option for selecting themes with predefined layouts. In addition to the `standard` which has the traditional `htmlTable` look, you also have `Google docs` and `blank`. The *Google docs* is still a work in progress and any help making it as compatible as possible with Google's Drive document when copy-pasting is much appreciated.

Using tidyverse syntax in tidyHtmlTable

In 2017, Stephen Gragg added the `tidyHtmlTable` to the package. Since then advances to RStudio has impacted in how we use R and it became obvious that the function should instead of strings as arguments directly accept column names just as defined by tidyselect. The `tidyHtmlTable` solves one of `htmlTable`'s greates weaknesses, the need for calculating the `rgroup`, `tspanner`, `cgroup` arguments and using it with the `tidyselect` interface is now pure joy:

```
library(tidyverse)
tribble(
  ~ rowname, ~ `No (First)`, ~ `% (First)`, ~ `No (Second)`, ~ `% (Second)`,
  "Group A",             20,             5,             380,             95,
  "Group B1",            11,            55,               9,             45,
  "Group B2",            11,            55,               9,             45,
) %>%
  pivot_longer(cols = c(starts_with("No"), starts_with("%"))) %>%
  mutate(group = str_replace(rowname, "Group ([AB]).*", "\\1"),
         rowname = str_replace(rowname, "Group ([AB12]+).*", "\\1"),
         group = if_else(group == rowname, "", group),
         header = str_replace(name, "([^ ]+).*", "\\1"),
         cgroup = str_replace(name, ".*\\(([^)]+)\\)$", "\\1")) %>%
  tidyHtmlTable(rgroup = group,
                header = header,
                cgroup = cgroup,
                rnames = rowname,
                caption = "A version of the first tables but using the tidyHtmlTable")
```

	First		Second	
	%	No	%	No
A	5	20	95	380
B				
B1	55	11	45	9
B2	55	11	45	9

Table 3: A version of the first tables but using the tidyHtmlTable

A more advanced example on how `tidyHtmlTable` works with `tidyverse` we can have a look at the example in

the `vignette("tidyHtmlTable")`:

```
mtcars %>%
  as_tibble(rownames = "rnames") %>%
  pivot_longer(names_to = "per_metric",
               cols = c(hp, mpg, qsec)) %>%
  group_by(cyl, gear, per_metric) %>%
  summarise(Mean = round(mean(value), 1),
            SD = round(sd(value), 1),
            Min = round(min(value), 1),
            Max = round(max(value), 1),
            .groups = 'drop') %>%
  pivot_longer(names_to = "summary_stat",
               cols = c(Mean, SD, Min, Max)) %>%
  ungroup() %>%
  mutate(gear = paste(gear, "Gears"),
         cyl = paste(cyl, "Cylinders")) %>%
  arrange(per_metric, summary_stat) %>%
  addHtmlTableStyle(align = "r") %>%
  tidyHtmlTable(header = gear,
                cgroup = cyl,
                rnames = summary_stat,
                rgroup = per_metric,
                caption = "A full example of how to apply the tidyverse workflow to
generate a table")
```

	4 Cylinders			6 Cylinders			8 Cylinders	
	3 Gears	4 Gears	5 Gears	3 Gears	4 Gears	5 Gears	3 Gears	5 Gears
hp								
Max	97	109	113	110	123	175	245	335
Mean	97	76	102	107.5	116.5	175	194.2	299.5
Min	97	52	91	105	110	175	150	264
SD		20.1	15.6	3.5	7.5		33.4	50.2
mpg								
Max	21.5	33.9	30.4	21.4	21	19.7	19.2	15.8
Mean	21.5	26.9	28.2	19.8	19.8	19.7	15.1	15.4
Min	21.5	21.4	26	18.1	17.8	19.7	10.4	15
SD		4.8	3.1	2.3	1.6		2.8	0.6
qsec								
Max	20	22.9	16.9	20.2	18.9	15.5	18	14.6
Mean	20	19.6	16.8	19.8	17.7	15.5	17.1	14.6
Min	20	18.5	16.7	19.4	16.5	15.5	15.4	14.5
SD		1.5	0.1	0.6	1.1		0.8	0.1

Table 4: A full example of how to apply the tidyverse workflow to generate a table

When using `tidyHtmlTable` you can decouple it from `htmlTable` and provide any table function by supplying the `table_fn` function.

Options

In htmlTable 2.0 there are plenty of options that have been added. Most of them should start with the prefix "htmlTable.", e.g "htmlTable.css.tspanner.sep". While the prefix is useful for reducing the risk of conflicting options between packages, options such as "table_counter" are unchanged in order to avoid unnecessary breaking changes.

NEWS for 2.0

- Added theming and styling with `addHtmlTableStyle` and `setHtmlTableTheme` to reduce the cognitive burden of finding the right option within the docs. Note: this may impact your current tables and hence the major version (2.0.0).

- Changed so that `css.cell` is properly applied to `rownames`, cell fillers and the actual cells of interest (may impact the final layout!)
- Breaking change `tidyHtmlTable`: Moved to a fully tidyverse compatible system with `tidyHtmlTable`. This is a breaking change to the API as we switch from columns as strings to `tidyselect` syntax and as `gather`/`spread` have been replaced by `pivot_longer`/`pivot_wider` the default values have been updated in accordance with their defaults, e.g. `rnames = "name"` and `value = "value"`.
- Breaking change `tidyHtmlTable`: Sorting of rows is skipped as we may have situations with repeating inputs and this can easily be performed pre-function by calling `dplyr::arrange`. This has furthermore the desirable feature that any custom sorting is retained.
- Added mso-number-format to help (Issue #63) – thanks Rasmus Hertzum
- `txtRound` can now add `txtInt` when formatting the integer section for easier readability
- Added `htmlTable` css options – they should all start with `htmlTable.`
- `pos.caption` now uses match.arg as expected
- Fixed proper S3 function definition for htmlTable with all the arguments
- Added `htmlTable.css.border` style option for allowing to choose border style. Also fixed bug with `cgroup` empty cells and vertical border.
- Added `htmlTable.pretty_indentation` option for skipping the stripping of all the tabs that was required due to old Pandoc bug.
- Added `attr(x, "html") <- TRUE` by default and UTF-8 encoding on all outputted strings to mimic the `htmltools::HTML` function behavior.
- For simple `tibble` output the `tidyHtmlTable` can now be used to choose a column for the `rnames` argument
- The `print` statement now respects the `chunk_output_type` in Rmd files in RStudio