Using the table1 Package to Create HTML Tables of Descriptive Statistics

Benjamin Rich

2021-06-06

Introduction

It is standard practice in epidemiology and related fields that the first table of any journal article, referred to as "Table 1", is a table that presents descriptive statistics of baseline characteristics of the study population stratified by exposure. This package makes it fairly straightforward to produce such a table using R. The output format is HTML (which has the advantage of being easy to copy into a Word document; Chrome browser works well). It is convenient to use this package in conjunction with knitr and R Markdown, as the HTML output is passed through untouched (note: as of version 1.1 it is no longer necessary to specify the `results='asis'` chunk option to have the HTML output appear correctly in the final document); in fact, this vignette serves as an example. The package does allow quite a bit of flexibility to customize the table's contents and appearance, but this does come at the cost of ease-of-use (more programming, some knowledge of CSS).

Example 1

The first example is inspired by this blog post, which is about how to accomplish a similar task using the `htmlTable` package. It uses the `melanoma` data set from the `boot` package for illustration, and I have copied here the code used to prepare the data:

```
library(boot)

melanoma2 <- melanoma

# Factor the basic variables that
# we're interested in
melanoma2$status <-
  factor(melanoma2$status,
         levels=c(2,1,3),
         labels=c("Alive", # Reference
                  "Melanoma death",
                  "Non-melanoma death"))
```

As a first attempt, we can do the following:

```
table1(~ factor(sex) + age + factor(ulcer) + thickness | status, data=melanoma2)
```

	Alive (N=134)	Melanoma death (N=57)	Non-melanoma death (N=14)	Overall (N=205)
factor(sex)				
0	91 (67.9%)	28 (49.1%)	7 (50.0%)	126 (61.5%)
1	43 (32.1%)	29 (50.9%)	7 (50.0%)	79 (38.5%)
age				
Mean (SD)	50.0 (15.9)	55.1 (17.9)	65.3 (10.9)	52.5 (16.7)

	Alive (N=134)	Melanoma death (N=57)	Non-melanoma death (N=14)	Overall (N=205)
Median [Min, Max]	52.0 [4.00, 84.0]	56.0 [14.0, 95.0]	65.0 [49.0, 86.0]	54.0 [4.00, 95.0]
factor(ulcer)				
0	92 (68.7%)	16 (28.1%)	7 (50.0%)	115 (56.1%)
1	42 (31.3%)	41 (71.9%)	7 (50.0%)	90 (43.9%)
thickness				
Mean (SD)	2.24 (2.33)	4.31 (3.57)	3.72 (3.63)	2.92 (2.96)
Median [Min, Max]	1.36 [0.100, 12.9]	3.54 [0.320, 17.4]	2.26 [0.160, 12.6]	1.94 [0.100, 17.4]

Note that the `table1` package uses a familiar formula interface, where the variables to include in the table are separated by '+' symbols, the "stratification" variable (which creates the columns) appears to the right of a "conditioning" symbol '|', and the `data` argument specifies a `data.frame` that contains the variables in the formula.

But because we don't have nice labels for the variables and categories, it doesn't look great. To improve things, we can create factors with descriptive labels for the categorical variables (`sex` and `ulcer`), label each variable the way we want, and specify units for the continuous variables (`age` and `thickness`), like this:

```
melanoma2$sex <- 
  factor(melanoma2$sex, levels=c(1,0),
         labels=c("Male", 
                  "Female"))

melanoma2$ulcer <- 
  factor(melanoma2$ulcer, levels=c(0,1),
         labels=c("Absent", 
                  "Present"))

label(melanoma2$sex)       <- "Sex"
label(melanoma2$age)       <- "Age"
label(melanoma2$ulcer)     <- "Ulceration"
label(melanoma2$thickness) <- "Thickness"

units(melanoma2$age)       <- "years"
units(melanoma2$thickness) <- "mm"

table1(~ sex + age + ulcer + thickness | status, data=melanoma2, overall="Total")
```

	Alive (N=134)	Melanoma death (N=57)	Non-melanoma death (N=14)	Total (N=205)
Sex				
Male	43 (32.1%)	29 (50.9%)	7 (50.0%)	79 (38.5%)
Female	91 (67.9%)	28 (49.1%)	7 (50.0%)	126 (61.5%)
Age (years)				
Mean (SD)	50.0 (15.9)	55.1 (17.9)	65.3 (10.9)	52.5 (16.7)
Median [Min, Max]	52.0 [4.00, 84.0]	56.0 [14.0, 95.0]	65.0 [49.0, 86.0]	54.0 [4.00, 95.0]
Ulceration				
Absent	92 (68.7%)	16 (28.1%)	7 (50.0%)	115 (56.1%)
Present	42 (31.3%)	41 (71.9%)	7 (50.0%)	90 (43.9%)
Thickness (mm)				
Mean (SD)	2.24 (2.33)	4.31 (3.57)	3.72 (3.63)	2.92 (2.96)
Median [Min, Max]	1.36 [0.100, 12.9]	3.54 [0.320, 17.4]	2.26 [0.160, 12.6]	1.94 [0.100, 17.4]

This looks better, but still not quite the same as the original blog post: in the blog post the "Total" column is on the left, while we have it on the right; the two "Death" strata (Melanoma and Non-melanoma) should be grouped together under a common heading; the continuous variables Age and Thickness show only Means (SD) (with a ±), and not Median [Min, Max] like the `table1` default output; most values are displayed with two significant digits rather than three (I will not concern myself with the footnote here, but it could be added as well). To achieve the same result, we need to customize the output further, and in this case that involves using the slightly more complicated "default" (i.e. non-formula) interface to `table1`.

First, we set up our labels differently, using a list:

```
labels <- list(
    variables=list(sex="Sex",
                   age="Age (years)",
                   ulcer="Ulceration",
                   thickness="Thickness (mm)"),
    groups=list("", "", "Death"))

# Remove the word "death" from the labels, since it now appears above
levels(melanoma2$status) <- c("Alive", "Melanoma", "Non-melanoma")
```

Next, we set up our “strata”, or column, as a list of `data.frame`s, in the order we want them displayed:

```
strata <- c(list(Total=melanoma2), split(melanoma2, melanoma2$status))
```

Finally, we can customize the contents using custom renderers. A custom render can be a function that take a vector as the first argument and return a (named) character vector. There is also a simpler way to customize the table contents using an abbreviated code syntax instead of a render function, but it allows less control over rounding (see below). Here, for example, we specify render functions for the continuous and categorical variables as follows:

```
my.render.cont <- function(x) {
    with(stats.apply.rounding(stats.default(x), digits=2), c("",
        "Mean (SD)"=sprintf("%s (&plusmn; %s)", MEAN, SD)))
}
my.render.cat <- function(x) {
    c("", sapply(stats.default(x), function(y) with(y,
        sprintf("%d (%0.0f %%)", FREQ, PCT))))
}
```

And here is the result:

```
table1(strata, labels, groupspan=c(1, 1, 2),
       render.continuous=my.render.cont, render.categorical=my.render.cat)
```

			Death	
	Total (N=205)	Alive (N=134)	Melanoma (N=57)	Non-melanoma (N=14)
Sex				
Male	79 (39 %)	43 (32 %)	29 (51 %)	7 (50 %)
Female	126 (61 %)	91 (68 %)	28 (49 %)	7 (50 %)
Age (years)				
Mean (SD)	52 (± 17)	50 (± 16)	55 (± 18)	65 (± 11)
Ulceration				
Absent	115 (56 %)	92 (69 %)	16 (28 %)	7 (50 %)
Present	90 (44 %)	42 (31 %)	41 (72 %)	7 (50 %)
Thickness (mm)				
Mean (SD)	2.9 (± 3.0)	2.2 (± 2.3)	4.3 (± 3.6)	3.7 (± 3.6)

This is now looking pretty similar to the original blog post, but admittedly there are still some differences: the sexes are inverted (the original blog post got it wrong); I added units to the continuous variables; I include the number of individuals in each column under the column heading; the percentages are different, because I think they should add to 100% within a column, and in the original blog post they add to 100% along a row (except for the Total column, which adds to 100% within the column). This last point is the most contentious. In my version, it is easier to compare the different types of outcomes with respect to variables like sex, while in the original version it is easier to compare sexes with respect to outcomes. However, this is not really the standard application for these kinds of tables (at least not the one I have in mind). Usually, the columns would represent exposure or treatment groups, not outcomes, and we want to compare those groups with respect to the distribution of baseline characteristics, and for this purpose having percentages add up to 100% within columns makes the most sense. Let’s continue with an example of that nature, using simulated data.

Example 2

For this second example, we will use simulated data. We imagine a clinical trial where subjects have been randomized in a 2:1 ratio to receive an active treatment or placebo. For simplicity, we will only consider three baseline characteristics: age, sex and weight.

```
f <- function(x, n, ...) factor(sample(x, n, replace=T, ...), levels=x)
set.seed(427)

n <- 146
dat <- data.frame(id=1:n)
dat$treat <- f(c("Placebo", "Treated"), n, prob=c(1, 2)) # 2:1 randomization
dat$age   <- sample(18:65, n, replace=TRUE)
dat$sex   <- f(c("Female", "Male"), n, prob=c(.6, .4))  # 60% female
dat$wt    <- round(exp(rnorm(n, log(70), 0.23)), 1)

# Add some missing data
dat$wt[sample.int(n, 5)] <- NA

label(dat$age)   <- "Age"
label(dat$sex)   <- "Sex"
label(dat$wt)    <- "Weight"
label(dat$treat) <- "Treatment Group"

units(dat$age)   <- "years"
units(dat$wt)    <- "kg"
```

Using the default settings, we obtain this table:

```
table1(~ age + sex + wt | treat, data=dat)
```

	Placebo (N=52)	**Treated (N=94)**	**Overall (N=146)**
Age (years)			
Mean (SD)	39.2 (14.2)	40.1 (13.3)	39.8 (13.6)
Median [Min, Max]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Sex			
Female	34 (65.4%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	41 (43.6%)	59 (40.4%)
Weight (kg)			
Mean (SD)	68.1 (16.3)	68.3 (16.7)	68.2 (16.5)
Median [Min, Max]	66.7 [37.5, 116]	64.9 [40.0, 119]	66.2 [37.5, 119]
Missing	2 (3.8%)	3 (3.2%)	5 (3.4%)

Note that when contains missing values (here weight), be it continuous or categorical, these are reported as a distinct category (with count and percent).

The “Overall” column can be easily removed (or relabeled):

```
table1(~ age + sex + wt | treat, data=dat, overall=F)
```

	Placebo (N=52)	**Treated (N=94)**
Age (years)		
Mean (SD)	39.2 (14.2)	40.1 (13.3)
Median [Min, Max]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]
Sex		
Female	34 (65.4%)	53 (56.4%)
Male	18 (34.6%)	41 (43.6%)
Weight (kg)		
Mean (SD)	68.1 (16.3)	68.3 (16.7)
Median [Min, Max]	66.7 [37.5, 116]	64.9 [40.0, 119]
Missing	2 (3.8%)	3 (3.2%)

We can also have stratification by two variables, in which case they are nested. For example, to see each treatment group split by sex:

```
table1(~ age + wt | treat*sex, data=dat)
```

	Placebo		Treated		Overall	
	Female (N=34)	**Male (N=18)**	**Female (N=53)**	**Male (N=41)**	**Female (N=87)**	**Male (N=59)**
Age (years)						
Mean (SD)	40.6 (14.5)	36.6 (13.6)	40.1 (13.4)	40.1 (13.3)	40.3 (13.8)	39.0 (13.4)
Median [Min, Max]	39.5 [18.0, 65.0]	33.5 [18.0, 64.0]	39.0 [18.0, 65.0]	41.0 [18.0, 65.0]	39.0 [18.0, 65.0]	39.0 [18.0, 65.0]
Weight (kg)						
Mean (SD)	68.8 (14.8)	66.8 (19.3)	65.6 (15.1)	71.5 (18.0)	66.9 (15.0)	70.1 (18.4)
Median [Min, Max]	67.2 [45.8, 116]	66.6 [37.5, 105]	61.4 [41.9, 103]	68.3 [40.0, 119]	63.8 [41.9, 116]	67.3 [37.5, 119]
Missing	1 (2.9%)	1 (5.6%)	3 (5.7%)	0 (0%)	4 (4.6%)	1 (1.7%)

Or, switch the order:

```
table1(~ age + wt | sex*treat, data=dat)
```

	Female		Male		Overall	
	Placebo (N=34)	**Treated (N=53)**	**Placebo (N=18)**	**Treated (N=41)**	**Placebo (N=52)**	**Treated (N=94)**
Age (years)						
Mean (SD)	40.6 (14.5)	40.1 (13.4)	36.6 (13.6)	40.1 (13.3)	39.2 (14.2)	40.1 (13.3)
Median [Min, Max]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]	33.5 [18.0, 64.0]	41.0 [18.0, 65.0]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]
Weight (kg)						
Mean (SD)	68.8 (14.8)	65.6 (15.1)	66.8 (19.3)	71.5 (18.0)	68.1 (16.3)	68.3 (16.7)
Median [Min, Max]	67.2 [45.8, 116]	61.4 [41.9, 103]	66.6 [37.5, 105]	68.3 [40.0, 119]	66.7 [37.5, 116]	64.9 [40.0, 119]
Missing	1 (2.9%)	3 (5.7%)	1 (5.6%)	0 (0%)	2 (3.8%)	3 (3.2%)

Or, no stratification:

```
table1(~ treat + age + sex + wt, data=dat)
```

	Overall (N=146)
Treatment Group	
Placebo	52 (35.6%)
Treated	94 (64.4%)
Age (years)	
Mean (SD)	39.8 (13.6)
Median [Min, Max]	39.0 [18.0, 65.0]
Sex	
Female	87 (59.6%)
Male	59 (40.4%)
Weight (kg)	
Mean (SD)	68.2 (16.5)
Median [Min, Max]	66.2 [37.5, 119]
Missing	5 (3.4%)

Finally, we may again consider something a bit more complicated, using the default (i.e., non-formula) interface. Suppose that instead of simply being assigned to placebo or active treatment, there were actually two doses of treatment randomized, 5 mg and 10 mg, and we want columns for each dose level separately, as well as for all treated subjects.

```
dat$dose <- (dat$treat != "Placebo")*sample(1:2, n, replace=T)
dat$dose <- factor(dat$dose, labels=c("Placebo", "5 mg", "10 mg"))
```

```r
strata <- c(split(dat, dat$dose), list("All treated"=subset(dat, treat=="Treated")), list(Overall=dat))

labels <- list(
    variables=list(age=render.varlabel(dat$age),
                   sex=render.varlabel(dat$sex),
                   wt=render.varlabel(dat$wt)),
    groups=list("", "Treated", ""))

table1(strata, labels, groupspan=c(1, 3, 1))
```

		Treated			
	Placebo (N=52)	**5 mg (N=43)**	**10 mg (N=51)**	**All treated (N=94)**	**Overall (N=146)**
Age (years)					
Mean (SD)	39.2 (14.2)	35.5 (12.4)	43.9 (12.9)	40.1 (13.3)	39.8 (13.6)
Median [Min, Max]	37.5 [18.0, 65.0]	37.0 [18.0, 64.0]	45.0 [21.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Sex					
Female	34 (65.4%)	24 (55.8%)	29 (56.9%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	19 (44.2%)	22 (43.1%)	41 (43.6%)	59 (40.4%)
Weight (kg)					
Mean (SD)	68.1 (16.3)	68.7 (16.3)	68.0 (17.1)	68.3 (16.7)	68.2 (16.5)
Median [Min, Max]	66.7 [37.5, 116]	67.2 [40.4, 111]	63.4 [40.0, 119]	64.9 [40.0, 119]	66.2 [37.5, 119]
Missing	2 (3.8%)	2 (4.7%)	1 (2.0%)	3 (3.2%)	5 (3.4%)

Using abbreviated code to specify a custom renderer

Suppose that for continuous variables, we want to display the percent coefficient of variation (CV%) instead of the standard deviation (SD). We also want to display the geometric mean and geometric coefficient of variation. We already discussed custom render functions that could be used to accomplish this, but a simpler alternative is to use abbreviated code. This is a character string that contains certain keywords which are substituted for computed values in the table output. The list of recognized keywords comes from the output of the `stats.default` function and includes: N, NMISS, MEAN, SD, CV, GMEAN, GCV, MEDIAN, MIN, MAX, IQR, Q1, Q2, Q3, T1, T2, FREQ, PCT. Keyword matching is case insensitive, and any text other than the keywords is left untouched. We can specify a vector of character strings, in which case each result will be displayed in its own row in the table. We can use a named vector to specify labels for each row; a dot ('.') can be used to indicate that the abbreviated code string itself be used as the row label. Significant digits can be controlled using the `digits` argument (default: 3). Here is a continuation of the example from the previous section that produces the desired result:

```r
table1(strata, labels, groupspan=c(1, 3, 1),
       render.continuous=c(.="Mean (CV%)", .="Median [Min, Max]",
                           "Geo. mean (Geo. CV%)"="GMEAN (GCV%)"))
```

		Treated			
	Placebo (N=52)	**5 mg (N=43)**	**10 mg (N=51)**	**All treated (N=94)**	**Overall (N=146)**
Age (years)					
Mean (CV%)	39.2 (36.2%)	35.5 (34.9%)	43.9 (29.4%)	40.1 (33.2%)	39.8 (34.2%)
Median [Min, Max]	37.5 [18.0, 65.0]	37.0 [18.0, 64.0]	45.0 [21.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Geo. mean (Geo. CV%)	36.7 (39.0%)	33.3 (38.1%)	41.9 (32.6%)	37.8 (37.1%)	37.4 (37.6%)
Sex					
Female	34 (65.4%)	24 (55.8%)	29 (56.9%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	19 (44.2%)	22 (43.1%)	41 (43.6%)	59 (40.4%)
Weight (kg)					
Mean (CV%)	68.1 (24.0%)	68.7 (23.7%)	68.0 (25.2%)	68.3 (24.4%)	68.2 (24.2%)
Median [Min, Max]	66.7 [37.5, 116]	67.2 [40.4, 111]	63.4 [40.0, 119]	64.9 [40.0, 119]	66.2 [37.5, 119]
Geo. mean (Geo. CV%)	66.2 (24.1%)	66.9 (23.5%)	66.0 (24.3%)	66.4 (23.8%)	66.4 (23.8%)
Missing	2 (3.8%)	2 (4.7%)	1 (2.0%)	3 (3.2%)	5 (3.4%)

Displaying different statistics for different variables

Suppose it is desired to show the median and range for age, but the mean and standard deviation for weight. This can be achieved using a custom render function as follows:

```r
rndr <- function(x, name, ...) {
    if (!is.numeric(x)) return(render.categorical.default(x))
    what <- switch(name,
        age = "Median [Min, Max]",
        wt  = "Mean (SD)")
    parse.abbrev.render.code(c("", what))(x)
}

table1(~ age + sex + wt | treat, data=dat,
       render=rndr)
```

	Placebo (N=52)	Treated (N=94)	Overall (N=146)
Age (years)			
Median [Min, Max]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Sex			
Female	34 (65.4%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	41 (43.6%)	59 (40.4%)
Weight (kg)			
Mean (SD)	68.1 (16.3)	68.3 (16.7)	68.2 (16.5)

Note that instead of overriding `render.continuous` and `render.categorical` separately, you can override `render` which handles both. The `render` function gets the name of the variable as its second argument, and should also accept `...` to capture any other arguments passed to it. Note also that the function `parse.abbrev.render.code` can be used to turn abbreviated code into a corresponding render function.

Changing the table's appearance

The default style of `table1` uses an Arial font, and resembles the booktabs style commonly used in LaTeX. While this default style is not ugly, inevitably there will be a desire to customize the visual appearance of the table (fonts, colors, gridlines, etc). The package provides a limited number of built-in options for changing the style, while further customization can be achieved in R Markdown documents using CSS (see below).

Using built-in styles

The package includes a limited number of built-in styles including:

- zebra: alternating shaded and unshaded rows (zebra stripes)
- grid: show all grid lines
- shade: shade the header row(s) in gray
- times: use a serif font
- center: center all columns, including the first which contains the row labels

These styles can be selected using the `topclass` argument of `table1`. Some examples follow:

```r
table1(~ age + sex + wt | treat, data=dat, topclass="Rtable1-zebra")
```

	Placebo (N=52)	Treated (N=94)	Overall (N=146)
Age (years)			
Mean (SD)	39.2 (14.2)	40.1 (13.3)	39.8 (13.6)
Median [Min, Max]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Sex			
Female	34 (65.4%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	41 (43.6%)	59 (40.4%)

	Placebo (N=52)	Treated (N=94)	Overall (N=146)
Weight (kg)			
Mean (SD)	68.1 (16.3)	68.3 (16.7)	68.2 (16.5)
Median [Min, Max]	66.7 [37.5, 116]	64.9 [40.0, 119]	66.2 [37.5, 119]
Missing	2 (3.8%)	3 (3.2%)	5 (3.4%)

```
table1(~ age + sex + wt | treat, data=dat, topclass="Rtable1-grid")
```

	Placebo (N=52)	Treated (N=94)	Overall (N=146)
Age (years)			
Mean (SD)	39.2 (14.2)	40.1 (13.3)	39.8 (13.6)
Median [Min, Max]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Sex			
Female	34 (65.4%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	41 (43.6%)	59 (40.4%)
Weight (kg)			
Mean (SD)	68.1 (16.3)	68.3 (16.7)	68.2 (16.5)
Median [Min, Max]	66.7 [37.5, 116]	64.9 [40.0, 119]	66.2 [37.5, 119]
Missing	2 (3.8%)	3 (3.2%)	5 (3.4%)

```
table1(~ age + sex + wt | treat, data=dat, topclass="Rtable1-grid Rtable1-shade Rtable1-times")
```

	Placebo (N=52)	Treated (N=94)	Overall (N=146)
Age (years)			
Mean (SD)	39.2 (14.2)	40.1 (13.3)	39.8 (13.6)
Median [Min, Max]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Sex			
Female	34 (65.4%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	41 (43.6%)	59 (40.4%)
Weight (kg)			
Mean (SD)	68.1 (16.3)	68.3 (16.7)	68.2 (16.5)
Median [Min, Max]	66.7 [37.5, 116]	64.9 [40.0, 119]	66.2 [37.5, 119]
Missing	2 (3.8%)	3 (3.2%)	5 (3.4%)

Note that the style name needs to be preceded by the prefix `Rtable1-`. Multiple styles can be applied in combination by separating them with a space.

Using custom CSS to control the table’s appearance

Further customization of the table appearance is only possible in R Markdown documents, by using custom CSS which is specified in the document’s YAML header. For examples, to include `style.css` in the output, the YAML header should contain the following:

```
output:
  html_document:
    css: style.css
```

CSS allows fine control of the appearance of different elements in the table. For examples, if `style.css` contains the following definitions:

```
table.Rtable1 {
    font-family: "Lucida Console", Monaco, monospace;
    border-collapse: collapse;
    font-size: 9pt;
}
.Rtable1 th {
    background-color: rgb(0, 100, 164);
    color: white;
}
.Rtable1 .firstrow, .Rtable1 .firstrow ~ td {
    border-top: 1pt solid black;
}
.Rtable1 td.rowlabel {
    color: DarkCyan;
    font-style: italic;
}
.Rtable1 td.firstrow.rowlabel {
    background-color: yellow;
    color: red;
    font-size: 12pt;
}
```

then the output will be as follows:

	Placebo (N=52)	Treated (N=94)	Overall (N=146)
Age (years)			
Mean (SD)	39.2 (14.2)	40.1 (13.3)	39.8 (13.6)
Median [Min, Max]	37.5 [18.0, 65.0]	39.5 [18.0, 65.0]	39.0 [18.0, 65.0]
Sex			
Female	34 (65.4%)	53 (56.4%)	87 (59.6%)
Male	18 (34.6%)	41 (43.6%)	59 (40.4%)
Weight (kg)			
Mean (SD)	68.1 (16.3)	68.3 (16.7)	68.2 (16.5)
Median [Min, Max]	66.7 [37.5, 116]	64.9 [40.0, 119]	66.2 [37.5, 119]
Missing	2 (3.8%)	3 (3.2%)	5 (3.4%)

(Note: as an alternative to redefining the default CSS class `Rtable1`, a different custom CSS class name could be used, and the `topclass` argument used to select it.)

Extra columns

Sometimes, it may be desired to add extra columns to the table, other than descriptive statistics. This can be accomplished using the `extra.col` option. The contents of the extra columns can be anything that can be computed from the data, making this an extremely flexible approach. As usual, this flexibility comes at a cost, namely in the form of more effort/code to achieve the desired result.

Example: a column of p-values

A user asked if it was possible to add a column to the table showing the p-value associated with a univariate test for differences in each variable across strata. This can be accomplished using the `extra.col` feature.

The following example uses the `lalonde` data from the `MatchIt` package. The dataset has a column `treat` that contains the value 0 for “Treatment” and 1 for “Control”; this will be used for stratification. In this example, a chi-square test of independence is used for categorical variables, and a t-test for continuous variables (other tests could be used if desired, this is just for illustration purposes).

First, we will assign factor levels, labels and units to the variables of interest.

```
library(MatchIt)
data(lalonde)

lalonde$treat    <- factor(lalonde$treat, levels=c(0, 1), labels=c("Control", "Treatment"))
lalonde$married  <- as.logical(lalonde$married == 1)
lalonde$nodegree <- as.logical(lalonde$nodegree == 1)
lalonde$race     <- factor(lalonde$race, levels=c("white", "black", "hispan"),
                                         labels=c("White", "Black", "Hispanic"))

label(lalonde$race)     <- "Race"
label(lalonde$married)  <- "Married"
label(lalonde$nodegree) <- "No high school diploma"
label(lalonde$age)      <- "Age"
label(lalonde$re74)     <- "1974 Income"
label(lalonde$re75)     <- "1975 Income"
label(lalonde$re78)     <- "1978 Income"
units(lalonde$age)      <- "years"
```

Next, we create a function to compute the p-value for continuous or categorical variables.

```
pvalue <- function(x, ...) {
    # Construct vectors of data y, and groups (strata) g
    y <- unlist(x)
    g <- factor(rep(1:length(x), times=sapply(x, length)))
    if (is.numeric(y)) {
        # For numeric variables, perform a standard 2-sample t-test
        p <- t.test(y ~ g)$p.value
    } else {
        # For categorical variables, perform a chi-squared test of independence
        p <- chisq.test(table(y, g))$p.value
    }
    # Format the p-value, using an HTML entity for the less-than sign.
    # The initial empty string places the output on the line below the variable label.
    c("", sub("<", "&lt;", format.pval(p, digits=3, eps=0.001)))
}
```

Note that this function expects a specific input, namely a list with 2 components corresponding to the 2 strata “Treatment” and “Control” in the `lalonde` data. These are the only 2 elements the list will have, because we will use `overall=F` in `table1` (otherwise, there would be a third element in the list corresponding to the overall column). Thus, this function is in some sense specifically tailored to this examples, and would need to be adapted to other situations (such as more than 2 strata, where a t-test would not work).

Now, we supply our function in the `extra.col` list argument to `table1` with the name `P-value`, which will appear as the column label (heading).

```
table1(~ age + race + married + nodegree + re74 + re75 + re78 | treat,
    data=lalonde, overall=F, extra.col=list(`P-value`=pvalue))
```

	Control (N=429)	**Treatment (N=185)**	**P-value**
Age (years)			
Mean (SD)	28.0 (10.8)	25.8 (7.16)	0.00291
Median [Min, Max]	25.0 [16.0, 55.0]	25.0 [17.0, 48.0]	
Race			
White	281 (65.5%)	18 (9.7%)	<0.001
Black	87 (20.3%)	156 (84.3%)	
Hispanic	61 (14.2%)	11 (5.9%)	
Married			
Yes	220 (51.3%)	35 (18.9%)	<0.001
No	209 (48.7%)	150 (81.1%)	
No high school diploma			
Yes	256 (59.7%)	131 (70.8%)	0.0113
No	173 (40.3%)	54 (29.2%)	
1974 Income			
Mean (SD)	5620 (6790)	2100 (4890)	<0.001
Median [Min, Max]	2550 [0, 25900]	0 [0, 35000]	
1975 Income			
Mean (SD)	2470 (3290)	1530 (3220)	0.00115
Median [Min, Max]	1090 [0, 18300]	0 [0, 25100]	
1978 Income			
Mean (SD)	6980 (7290)	6350 (7870)	0.349
Median [Min, Max]	4980 [0, 25600]	4230 [0, 60300]	

Admittedly, this is not as simple as setting a flag, but has the advantage of being totally flexible.

Transposed table

By default, the table produced by `table1` will have strata or subgroups as columns, and variables as rows. In some cases, it may be desirable to transpose the table such that each column is a variables and the rows are strata. This makes most sense when all the variables are continuous and when a compact representation is desired. It can be achieved by using the `transpose = TRUE` option.

An example:

```
dat <- expand.grid(i=1:50, group=LETTERS[1:3])
dat <- cbind(dat, matrix(round(exp(rnorm(6*nrow(dat))), 1), nrow=nrow(dat)))
names(dat)[3:8] <- paste0("V", 1:6)
```

Default:

```
table1(~ V1 + V2 + V3 + V4 + V5 + V6 | group, data=dat,
       topclass="Rtable1-grid Rtable1-center",
       render="Mean (CV%)<br/>Median [Min, Max]<br/>GMean (GCV%)")
```

	A **(N=50)**	**B** **(N=50)**	**C** **(N=50)**	**Overall** **(N=150)**
V1	1.60 (122.5%) 1.15 [0.100, 12.7] 0.968 (148.9%)	1.62 (152.1%) 1.00 [0.100, 13.6] 0.829 (169.3%)	1.39 (141.5%) 1.00 [0.200, 13.4] 0.884 (113.7%)	1.54 (138.7%) 1.00 [0.100, 13.6] 0.892 (142.2%)
V2	1.31 (139.0%) 0.850 [0.100, 11.4] 0.778 (130.4%)	1.51 (102.3%) 0.900 [0.100, 7.00] 0.913 (146.1%)	1.48 (158.2%) 0.800 [0.100, 13.4] 0.834 (135.1%)	1.43 (134.0%) 0.900 [0.100, 13.4] 0.840 (136.1%)
V3	1.80 (115.5%) 1.10 [0.200, 8.90] 1.10 (125.2%)	2.55 (333.0%) 1.00 [0.100, 60.5] 1.00 (145.7%)	1.29 (93.6%) 1.05 [0.200, 7.40] 0.917 (101.6%)	1.88 (270.7%) 1.05 [0.100, 60.5] 1.01 (123.2%)
V4	1.44 (107.6%) 0.900 [0.200, 6.70] 0.956 (106.6%)	1.70 (150.8%) 1.20 [0.100, 18.0] 1.06 (121.8%)	1.97 (129.1%) 0.850 [0.100, 12.2] 1.04 (157.2%)	1.70 (132.9%) 0.900 [0.100, 18.0] 1.02 (127.0%)
V5	1.25 (105.2%) 0.800 [0.100, 7.70] 0.840 (112.3%)	1.94 (128.5%) 1.15 [0.100, 15.1] 1.15 (138.2%)	1.73 (80.3%) 1.40 [0.100, 7.10] 1.24 (113.0%)	1.64 (111.2%) 1.15 [0.100, 15.1] 1.06 (122.7%)
V6	1.87 (204.2%) 0.900 [0.100, 26.4] 0.925 (157.1%)	1.96 (152.9%) 1.00 [0.100, 19.9] 1.00 (174.5%)	2.06 (125.6%) 1.00 [0.100, 12.0] 1.13 (150.1%)	1.96 (160.8%) 1.00 [0.100, 26.4] 1.02 (159.3%)

Transposed:

```
table1(~ V1 + V2 + V3 + V4 + V5 + V6 | group, data=dat,
       topclass="Rtable1-grid Rtable1-center",
       render="Mean (CV%)<br/>Median [Min, Max]<br/>GMean (GCV%)",
       transpose=TRUE)
```

	V1	**V2**	**V3**	**V4**	**V5**	**V6**
A **(N=50)**	1.60 (122.5%) 1.15 [0.100, 12.7] 0.968 (148.9%)	1.31 (139.0%) 0.850 [0.100, 11.4] 0.778 (130.4%)	1.80 (115.5%) 1.10 [0.200, 8.90] 1.10 (125.2%)	1.44 (107.6%) 0.900 [0.200, 6.70] 0.956 (106.6%)	1.25 (105.2%) 0.800 [0.100, 7.70] 0.840 (112.3%)	1.87 (204.2%) 0.900 [0.100, 26.4] 0.925 (157.1%)
B **(N=50)**	1.62 (152.1%) 1.00 [0.100, 13.6] 0.829 (169.3%)	1.51 (102.3%) 0.900 [0.100, 7.00] 0.913 (146.1%)	2.55 (333.0%) 1.00 [0.100, 60.5] 1.00 (145.7%)	1.70 (150.8%) 1.20 [0.100, 18.0] 1.06 (121.8%)	1.94 (128.5%) 1.15 [0.100, 15.1] 1.15 (138.2%)	1.96 (152.9%) 1.00 [0.100, 19.9] 1.00 (174.5%)
C **(N=50)**	1.39 (141.5%) 1.00 [0.200, 13.4] 0.884 (113.7%)	1.48 (158.2%) 0.800 [0.100, 13.4] 0.834 (135.1%)	1.29 (93.6%) 1.05 [0.200, 7.40] 0.917 (101.6%)	1.97 (129.1%) 0.850 [0.100, 12.2] 1.04 (157.2%)	1.73 (80.3%) 1.40 [0.100, 7.10] 1.24 (113.0%)	2.06 (125.6%) 1.00 [0.100, 12.0] 1.13 (150.1%)
Overall **(N=150)**	1.54 (138.7%) 1.00 [0.100, 13.6] 0.892 (142.2%)	1.43 (134.0%) 0.900 [0.100, 13.4] 0.840 (136.1%)	1.88 (270.7%) 1.05 [0.100, 60.5] 1.01 (123.2%)	1.70 (132.9%) 0.900 [0.100, 18.0] 1.02 (127.0%)	1.64 (111.2%) 1.15 [0.100, 15.1] 1.06 (122.7%)	1.96 (160.8%) 1.00 [0.100, 26.4] 1.02 (159.3%)