Modifying atable

Alan Haynes, Armin Ströbel

February 18, 2022

Contents

Most things in ‘atable‘ are customizable. This vignette gives examples for some of them. See the ‘extending atable‘ vignette for adding methods for new classes. We will the ‘mtcars‘ datasets to demonstrate the concepts.

```
data(mtcars)
# factors
mtcars$am <- factor(mtcars$am, c(0, 1), c("Automatic", "Manual"))
mtcars$vs <- factor(mtcars$vs, c(0, 1), c("V-shaped", "straight"))
# ordered
mtcars$cyl <- ordered(mtcars$cyl)
# set format_to
atable_options(format_to = "Latex")
```

The atable default settings produce the following:

```
Hmisc::latex(atable(vs + cyl + hp + disp ~ am, mtcars, format_to="Latex"),
             file = "",
             title = "",
             rowname = NULL,
             table.env = FALSE)
```

Group	Automatic	Manual	p	stat	Effect Size (CI)
Observations					
	19	13			
vs					
V-shaped	63% (12)	46% (6)	0.56	0.35	2 (0.38; 11)
straight	37% (7)	54% (7)			
missing	0% (0)	0% (0)			
cyl					
4	16% (3)	62% (8)	0.0039	194	0.57 (0.18; 0.81)
6	21% (4)	23% (3)			
8	63% (12)	15% (2)			
missing	0% (0)	0% (0)			
hp					
Mean (SD)	160 (54)	127 (84)	0.038	0.51	0.49 (-0.25; 1.2)
valid (missing)	19 (0)	13 (0)			
disp					
Mean (SD)	290 (110)	144 (87)	0.0013	0.69	1.4 (0.62; 2.3)
valid (missing)	19 (0)	13 (0)			

There are in general three approaches to modifying atable's default settings:

1. pass a function to the atable function as an option. This affects only a single call to atable.
2. pass a function to 'atable_options'. This affects all calls to 'atable' during the session. These settings can be overriden by 1
3. replace a method in 'atable's namespace. This can be done with any package, but requires more code and is not easily reverted.

1 Changing the statistical functions

In order to use any of these methods, we need suitable functions. Perhaps it is desirable to have t-tests and Kolmogorov-Smirnoff tests simultaneously. When defining test functions, the function has to have arguments 'value', 'group' and '...' and return a named list.

```
new_two_sample_htest_numeric <- function(value, group, ...){
  d <- data.frame(value = value, group = group)
  group_levels <- levels(group)
  x <- subset(d, group %in% group_levels[1], select = "value", drop = TRUE)
  y <- subset(d, group %in% group_levels[2], select = "value", drop = TRUE)
  ks_test_out <- stats::ks.test(x, y)
  t_test_out <- stats::t.test(x, y)
```

```
  out <- list(p_ks = ks_test_out$p.value,
              p_t = t_test_out$p.value)
  return(out)
}
```

Rather than mean and SD, maybe we want median and MAD, as well as the mean and SD. Statistics functions require arguments 'x' and '...' and should return a named list. The class of the output should also be defined so that the appropriate formatting function can be selected.

```
new_statistics_numeric <- function(x, ...){
  statistics_out <- list(Median = median(x, na.rm = TRUE),
                         MAD = mad(x, na.rm = TRUE),
                         Mean = mean(x, na.rm = TRUE),
                         SD = sd(x, na.rm = TRUE))
  class(statistics_out) <- c("statistics_numeric", class(statistics_out))
  # We will need this new class later to specify the format
  return(statistics_out)
}
```

We also need a function to format the statistics results (the default simply prints all elements of the statistics object, one after the other). Formatting functions require arguments 'x' and '...' and should return a dataframe with variable tag (a factor) and value (a character).

```
new_format_statistics_numeric <- function(x, ...){
  Median_MAD <- paste(round(c(x$Median, x$MAD), digits = 1), collapse = "; ")
  Mean_SD <- paste(round(c(x$Mean, x$SD), digits = 1), collapse = "; ")
  levs <- c("Median; MAD", "Mean; SD")
  out <- data.frame(tag = factor(levs,
                                 levels = levs),
                    # the factor needs levels for the non-alphabetical order
                    value = c(Median_MAD, Mean_SD),
                    stringsAsFactors = FALSE)
  return(out)
}
```

To use these three functions, we need to tell 'atable' about them, using one of the three methods mentioned above. We will assign the testing function to 'atable's namespace. 'two_sample_htest.numeric' will be used for two sample tests until R is restarted.

```
utils::assignInNamespace(x = "two_sample_htest.numeric",
                         value = new_two_sample_htest_numeric,
                         ns = "atable")
```

'atable_options' can be used to specify a new function instead by referring to the thing to be replaced, in this case 'statistics.numeric'. The 'new_statistics_numeric' function will be used until e.g. R is restarted or 'atable_options_reset' is used to restore the defaults, or it is replaced by something else.

```
atable_options("statistics.numeric" = new_statistics_numeric)
```

The third option is to refer to the function in the 'atable' call...

```
Hmisc::latex(atable(hp + disp ~ am, mtcars,
                    format_statistics.statistics_numeric =
                      new_format_statistics_numeric),
             file = "",
             title = "",
             rowname = NULL,
             table.env = FALSE)
```

Group	Automatic	Manual	p_ks	p_t
Observations				
	19	13		
hp				
Median; MAD	175; 77.1	109; 63.8	0.038	0.22
Mean; SD	160.3; 53.9	126.8; 84.1		
disp				
Median; MAD	275.8; 124.8	120.3; 58.9	0.0013	2.3e-04
Mean; SD	290.4; 110.2	143.5; 87.2		

Options that have been set via 'atable_options' can be restored to there defaults via

```
atable_options_reset()
```

2 Labels

'atable' can use Hmisc::label and Hmisc::unit, or an 'alias' attribute to store labels. By default 'labelled' objects (those with labels from Hmisc) are formatted with the label followed by the units in square brackets.

```
label(mtcars$hp) <- "Horse power"
units(mtcars$hp) <- "hp"
```

```
Hmisc::latex(atable(hp + disp ~ 1, mtcars),
             file = "",
             title = "",
             rowname = NULL,
             table.env = FALSE)
```

Group	value
Observations	
	32
Horse power [hp]	
Mean (SD)	147 (69)
valid (missing)	32 (0)
disp	
Mean (SD)	231 (124)
valid (missing)	32 (0)

Analagous to elsewhere, the 'get_alias' methods are responsible for handling how the label is composed. If there is no 'alias' attribute and the class is not 'labelled', then the variable name will be used, with underscore replaced by blanks. We can use round brackets instead of square ones by replacing the 'get_alias.labelled' method...

```
get_alias.labelled <- function(x, ...){
    out <- attr(x, "label", exact = TRUE)
    Units <- attr(x, "units", exact = TRUE)
    out = if(!is.null(Units)){
      paste0(out, " (", Units, ")")}else{out}
    return(out)
}
atable_options("get_alias.labelled" = get_alias.labelled)
```

```
Hmisc::latex(atable(hp + disp ~ 1, mtcars),
             file = "",
             title = "",
             rowname = NULL,
             table.env = FALSE)
```

Group	value
Observations	
	32
Horse power (hp)	
Mean (SD)	147 (69)
valid (missing)	32 (0)
disp	
Mean (SD)	231 (124)
valid (missing)	32 (0)

If we wanted to use a 'label' attribute rather than 'alias', we could change the 'get_alias.default' function

```
attr(mtcars$disp, "label") <- "Displacement"
get_alias.default <- function(x, ...){
    attr(x, "label", exact = TRUE)
}
atable_options("get_alias.default" = get_alias.default)
```

```
Hmisc::latex(atable(hp + disp ~ 1, mtcars),
             file = "",
             title = "",
             rowname = NULL,
             table.env = FALSE)
```

Group	value
Observations	
	32
Horse power (hp)	
Mean (SD)	147 (69)
valid (missing)	32 (0)
Displacement	
Mean (SD)	231 (124)
valid (missing)	32 (0)

3 P value formatting

By default, p values are formatted to 2 significant digits:

```
atable_options("format_p_values")(0.12)

## [1] "0.12"
```

```
atable_options("format_p_values")(0.012)

## [1] "0.012"

atable_options("format_p_values")(0.0012)

## [1] "0.0012"

atable_options("format_p_values")(0.0009)

## [1] "<0.001"
```

This is easy to change by overwriting the function in 'atable_options'. Here we report it to 3 decimal places.

```
fn <- function(x){
  txt <- sprintf("%3.3f", x)
  if(x < 0.001) txt <- "<0.001"
  return(txt)
}
atable_options("format_p_values" = fn)
```

```
atable_options("format_p_values")(0.12)

## [1] "0.120"

atable_options("format_p_values")(0.012)

## [1] "0.012"

atable_options("format_p_values")(0.0012)

## [1] "0.001"

atable_options("format_p_values")(0.0009)

## [1] "<0.001"
```

The changes are then carried over to the table itself too.

```
Hmisc::latex(atable(vs + cyl + hp + disp ~ am, mtcars),
             file = "",
             title = "",
             rowname = NULL,
             table.env = FALSE)
```

Group	Automatic	Manual	p	stat	Effect Size (CI)	p_ks	p_t
Observations							
	19	13					
vs							
V-shaped	63% (12)	46% (6)	0.556	0.35	2 (0.38; 11)		
straight	37% (7)	54% (7)					
missing	0% (0)	0% (0)					
cyl							
4	16% (3)	62% (8)	0.004	194	0.57 (0.18; 0.81)		
6	21% (4)	23% (3)					
8	63% (12)	15% (2)					
missing	0% (0)	0% (0)					
Horse power (hp)							
Mean (SD)	160 (54)	127 (84)				0.038	0.22
valid (missing)	19 (0)	13 (0)					
Displacement							
Mean (SD)	290 (110)	144 (87)				0.0013	2.3e-04
valid (missing)	19 (0)	13 (0)					

It should be noted that as there is no additional formatting function for the test results, so the default method is being used for the new. We would have to assign a class to the object returned from ‘new_two_sample_htest_numeric‘ and define a suitable test formatting function.