R, Latex and RMarkdown: table() input for xtable() missing column and row labels?

Benjamin Nutter

2016-04-11

```
library(xtable)
```

The Question

R, Latex and RMarkdown: table() input for xtable() missing column and row labels?

I am trying to create a pretty LaTeX table where the names of the row and column variables of a table are included when using the xtable library.

MWE:

```
test <- data.frame(Apples=c(1,2,3), Oranges=c(4,5,6), Watermelons=c(7,8,9))
testxtab <- xtable(with(test,table(Apples,Oranges)))
print(testxtab, comment=FALSE)
```

	4	5	6
1	1	0	0
2	0	1	0
3	0	0	1

The result is a LaTeX table that is missing the "Apples" and "Oranges" labels. How do I include them?

`pixiedust` Solution

The user wants a solution that returns a table that appears as

```
with(test, table(Apples, Oranges))
```

```
##       Oranges
## Apples 4 5 6
##      1 1 0 0
##      2 0 1 0
##      3 0 0 1
```

In `pixiedust`, we can interpret this as a table with four columns, and a header with two rows. The real challenge in getting this format is that the `Oranges` and `Apples` labels are not easily retained from the `table` object.

As a matter of a two dimensional table, this isn't an insurmountable task (perhaps a bit tedious).

	Oranges		
Apples	4	5	6
1	1	0	0
2	0	1	0
3	0	0	1

```
library(pixiedust)
obj <- with(test, table(Apples, Oranges))

dimnames <- names(attr(obj, "dimnames"))

head <-
  rbind(c("", dimnames[2], rep("", ncol(obj) - 1)),
        c(dimnames[1], colnames(obj))) %>%
  as.data.frame(stringsAsFactors = FALSE)
body <- cbind(rownames(obj), obj) %>%
  as.data.frame(stringsAsFactor = FALSE)

dust(body) %>%
  redust(head, part = "head") %>%
  sprinkle(rows = 1,
           cols = 2:4,
           merge = TRUE,
           part = "head") %>%
  medley_bw() %>%
  sprinkle_print_method("latex")
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

The issue here is that the solution doesn't generalize beyond a two-dimensional table. What we need is a solution that can accommodate an n-dimensional table. Let's try a more general approach now. In this approach, we'll take advantage of the list methods for `dust` and `sprinkle`. All we need to do is get the `table` object into a list.

First, `broom::tidy` does a good job of restructuring the object into a tidy data frame.

```
library(broom)
library(pixiedust)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)

obj <- with(test, table(Apples, Oranges, Watermelons))
tidy_obj <- tidy(obj)
```

Next, we observe that the first two columns of the tidy data frame combine to make a 2x2 table, and the last column gives the frequency of each combination of the dimensions. It follows that columns 3 through `n-1` are stratifying variables. So why not split the data frame on those?

```
split_obj <- split(tidy_obj, tidy_obj[-c(1, 2, ncol(tidy_obj))])
```

Next, we need to get the data frames back into the two dimensional tables. This is a good task for `tidyr::spread_` (we'll use the underscore version so we can use character vectors). Before we do our `spread_`, we'll want to drop any of the stratifying variables.

```
split_obj <-
  lapply(split_obj,
         function(x) x[, c(1, 2, ncol(x))])

split_obj <-
  lapply(split_obj,
         function(x) spread_(x, names(x)[2], names(x)[3]))
```

At this point, we have a list of three tables, each of which can be formatted using the same process we used earlier. Only this time, we need to decide what to do with the extra dimension information.

```
head <-
  rbind(c("", dimnames[2], rep("", ncol(obj) - 1)),
        c(dimnames[1], colnames(obj))) %>%
  as.data.frame(stringsAsFactors = FALSE)

latex_tables <-
  dust(split_obj,
       longtable = TRUE) %>%
  redust(head, part = "head")  %>%
  sprinkle(rows = 1,
           cols = 2:4,
           merge = TRUE,
           part = "head") %>%
  medley_bw() %>%
  sprinkle_print_method("latex")
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped

## Warning: attributes are not identical across measure variables; they will
## be dropped

## Warning: attributes are not identical across measure variables; they will
## be dropped

## Warning in max(x$body$row): no non-missing arguments to max; returning -Inf
```

The only thing left to do now is to put the remaining dimension information into the table somewhere. For this purpose, I'll put it into the caption.

```
Strata <-
  select_(tidy_obj,
          .dots = names(tidy_obj)[-c(1, 2, ncol(tidy_obj))]) %>%
  distinct_(.dots = names(.))

captions <-
  lapply(names(Strata),
       function(x) sprintf("%s = %s", x, Strata[[x]])) %>%
  as.data.frame(stringsAsFactors = FALSE) %>%
  apply(., MARGIN = 1, FUN = paste, collapse = "; ")


for(i in seq_along(latex_tables))
{
  latex_tables[[i]] <- sprinkle(latex_tables[[i]],
                                caption = captions[i])
}

latex_tables
```

Table 1: Watermelons = 7

	Oranges		
Apples	4	5	6
1	1	0	0
2	0	0	0
3	0	0	0

Table 2: Watermelons = 8

	Oranges		
Apples	4	5	6
1	0	0	0
2	0	1	0
3	0	0	0

Table 3: Watermelons = 9

	Oranges		
Apples	4	5	6
1	0	0	0
2	0	0	0
3	0	0	1

And there's no reason this couldn't be a function

```
ndim_table <- function(object, print_method = "latex")
{
  require(broom)
  require(dplyr)
  require(pixiedust)
  require(tidyr)
  tidy_object <- broom::tidy(object)

  split_object <-
    split(tidy_object,
          tidy_object[-c(1, 2, ncol(tidy_object))]) %>%
    lapply(.,
           function(x) x[, c(1, 2, ncol(x))]) %>%
    lapply(.,
           function(x) tidyr::spread_(x, names(x)[2], names(x)[3]))

  head <-
    rbind(c("", names(dimnames(object))[2], rep("", ncol(object) - 1)),
          c(names(dimnames(object))[1], colnames(object))) %>%
    as.data.frame(stringsAsFactors = FALSE)

  output_tables <-
    pixiedust::dust(split_object,
         longtable = TRUE) %>%
    pixiedust::redust(head, part = "head")  %>%
    pixiedust::sprinkle(rows = 1,
                        cols = 2:4,
                        merge = TRUE,
                        part = "head") %>%
    pixiedust::medley_bw() %>%
    pixiedust::sprinkle_print_method(print_method)

  Strata <-
  dplyr::select_(tidy_object,
                 .dots = names(tidy_object)[-c(1, 2, ncol(tidy_object))]) %>%
  dplyr::distinct_(.dots = names(.))

  captions <-
    lapply(names(Strata),
        function(x) sprintf("%s = %s", x, Strata[[x]])) %>%
    as.data.frame(stringsAsFactors = FALSE) %>%
    apply(., MARGIN = 1, FUN = paste, collapse = "; ")

  for(i in seq_along(latex_tables))
  {
    output_tables[[i]] <- sprinkle(output_tables[[i]],
                                   caption = captions[i])
  }

  output_tables
}

ndim_table(with(test, table(Apples, Oranges, Watermelons)))
```

Table 4: Watermelons = 7

	Oranges		
Apples	4	5	6
1	1	0	0
2	0	0	0
3	0	0	0

Table 5: Watermelons = 8

	Oranges		
Apples	4	5	6
1	0	0	0
2	0	1	0
3	0	0	0

Table 6: Watermelons = 9

	Oranges		
Apples	4	5	6
1	0	0	0
2	0	0	0
3	0	0	1

```
#* Just to put in an extra dimension.
ndim_table(with(mtcars, table(gear, carb, vs, am)))
```

Table 7: vs = 0; am = 0

	carb					
gear	1	2	3	4	6	8
3	0	4	3	5	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Table 8: vs = 1; am = 0

	carb					
gear	1	2	3	4	6	8
3	3	0	0	0	0	0
4	0	2	0	2	0	0
5	0	0	0	0	0	0

Table 9: vs = 0; am = 1

	carb					
gear	1	2	3	4	6	8
3	0	0	0	0	0	0
4	0	0	0	2	0	0
5	0	1	0	1	1	1

	carb					
gear	1	2	3	4	6	8
3	0	0	0	0	0	0
4	4	2	0	0	0	0
5	0	1	0	0	0	0