

Labeling in Stucplots

by David Meyer
Wirtschaftsuniversität Wien, Austria
David.Meyer@R-project.org

August 2, 2005

One of the major enhancements in `vcd` is the labeling of strucplots which now has become much more powerful and flexible. Like the panel, the shading, the spacing, and the legend, labeling is now completely modular. The user supplies either a labeling function, or, alternatively, a generating function that parameterizes a labeling function, to `strucplot()` which then draws the labels. The labeling is well-separated from the actual plotting that occurs in the panel functions; it actually only relies on the viewport tree produced by them, and the ‘dimnames’ attribute of the visualized table. The drawing of the labels happens after the panel function has drawn the actual plot. Thus, it is possible to supply one’s own labeling function, or to combine some of the basic functions to produce a more complex labeling. As an example, consider `labeling_list()`:

```
labeling_list <- function(gp = gpar(), just = "left", pos = "left", lsep = ": ",
                        sep = " ", offset = unit(c(2, 2), "lines"),
                        varnames = TRUE, cols = 2, ...)
  function(d, split_vertical, condvars) {
    \dots
  }
```

`labeling_list()` is a generating function that takes all parameters as arguments, and returns the actual labeling function which takes only three arguments: the dimnames structure of the table, the logical vector with the splitting information, and the number of conditioning variables (if any). The labeling function has access to the arguments of the generating function because of lexical scoping. In the following, we describe the three basic modules (`labeling_text()`, `labeling_list()`, and `labeling_cells()`) and derived functions that build upon them.

1 Labels in the borders: `labeling_text()`

`labeling_text()` is the default for all strucplot displays. It plots labels in the borders similar to the `mosaicplot()` function, but is much more flexible: it is not limited to 4 dimensions, and the positioning and graphical parameters of levels and variable names are quite customizable. In addition, the problem of overlapping labels can be handled in several ways.

As an example, consider the ‘Survival on the Titanic’ data, consisting of 4 categorical variables: Survival, Gender (“Sex”), Age, and Crew. By default, the variable names and levels are plotted ‘around’ the plot in a counter-clockwise way (see Figure 1):

```
> mosaic(Titanic)
```

Note that the last two levels of the ‘Survived’ variable do overlap. This can be addressed in several ways. The ‘brute force’ method is to enable clipping for the ‘Survival’ dimension (see Figure 2):

```
> mosaic(Titanic, labeling_args = list(clip = c(Survived = TRUE)))
```

A more sensible solution is to abbreviate the ‘Survival’ levels (see Figure 3):

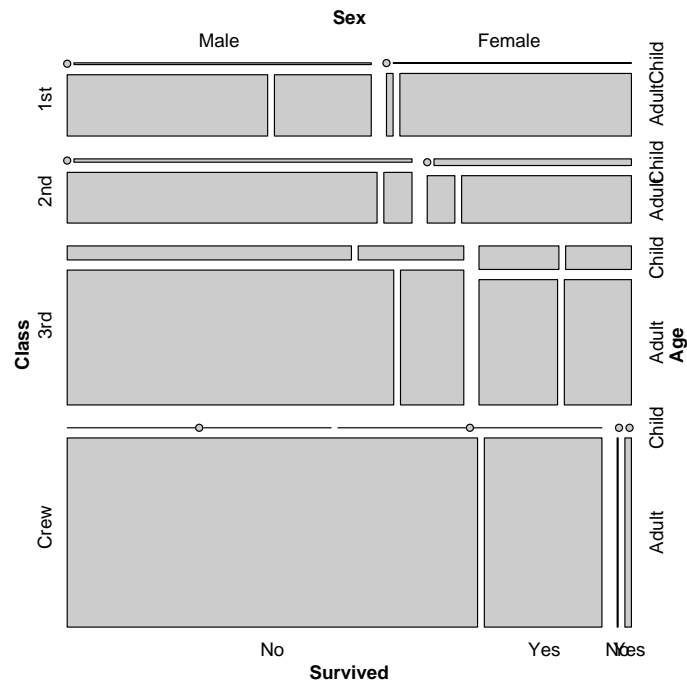


Figure 1: Mosaic plot for the Titanic data.

```
> mosaic(Titanic, labeling_args = list(abbreviate = c(Survived = TRUE)))
```

The `abbreviate` argument actually takes a vector of integers indicating the number of characters the levels should be abbreviated to, but logical values are equivalent to 1. Another possibility is to rotate the levels (see Figure 4):

```
> mosaic(Titanic, labeling_args = list(rot_labels = c(bottom = 45)))
```

Finally, we could also inhibit the output of repeated levels for the ‘Survival’ dimension (see Figure 5):

```
> mosaic(Titanic, labeling_args = list(rep = c(Survived = FALSE)))
```

We proceed with a few more ‘cosmetic’ changes. A first simple, but effectful modification is to position all labels and variables left-aligned, which is also achieved by using `labeling_left()` (see Figure 6):

```
> mosaic(Titanic, labeling_args = list(pos_varnames = "left", pos_labels = "left", just_labels =
```

Next, we show how to put all levels to the bottom and right margins, and all variable names to the top and left margins (see Figure 7):

```
> mosaic(Titanic, labeling_args = list(tl_labels = FALSE, tl_varnames = TRUE))
```

Now, we will add boxes to the labels and additionally enable clipping (see Figure 8):

```
> mosaic(Titanic, labeling_args = list(tl_labels = FALSE, tl_varnames = TRUE, boxes = TRUE, clip
```

This is pretty close to what the `labels_cboxed()` wrapper does. Another variant is to put the variable names into the same line as the levels (see Figure 9):

```
> mosaic(Titanic, labeling_args = list(tl_labels = TRUE, boxes = TRUE, clip = TRUE, labbl_varname
```

This is basically what `labeling_lboxed()` produces, and a similar design is used by the `dou-bledecker()`.

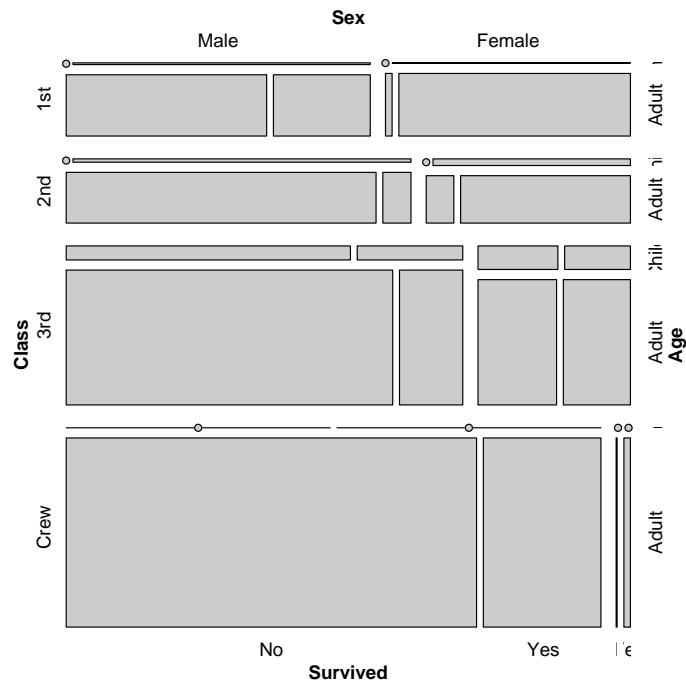


Figure 2: The effect of clipping.

2 Labels in the cells: `labeling_cells()`

This labeling draws both variable names and levels in the cells. As an example, we use the ‘PreSex’ data on pre- and extramarital sex and divorce (see Figure 10):

```
> mosaic(~ MaritalStatus + Gender, data = PreSex, labeling = labeling_cells)
```

In the case of narrow cells, the labels can be abbreviated (see Figure 11):

```
> mosaic(~ PremaritalSex + ExtramaritalSex, data = PreSex, labeling = labeling_cells(abbreviate_1))
```

For some data, it might be convenient to combine cell labeling with border labeling as done by `labels_conditional()` (see Figure 12):

```
> mosaic(~ PremaritalSex + ExtramaritalSex | MaritalStatus + Gender, data = PreSex, labeling = labels_conditional)
```

Additionally, the cell labeling allows the user to add arbitrary text to the cells by supplying a character array in the same shape than the data array to the `text` argument (cells with missing values are ignored). In the following example, this is used to add all observed values greater than 5 to the cells after the mosaic for the ‘Titanic’ data has been plotted (see Figure 13):

```
> mosaic(Titanic, pop = FALSE)

> tab <- ifelse(Titanic < 6, NA, Titanic)
> labeling_cells(text = tab, clip = FALSE)(Titanic)
```

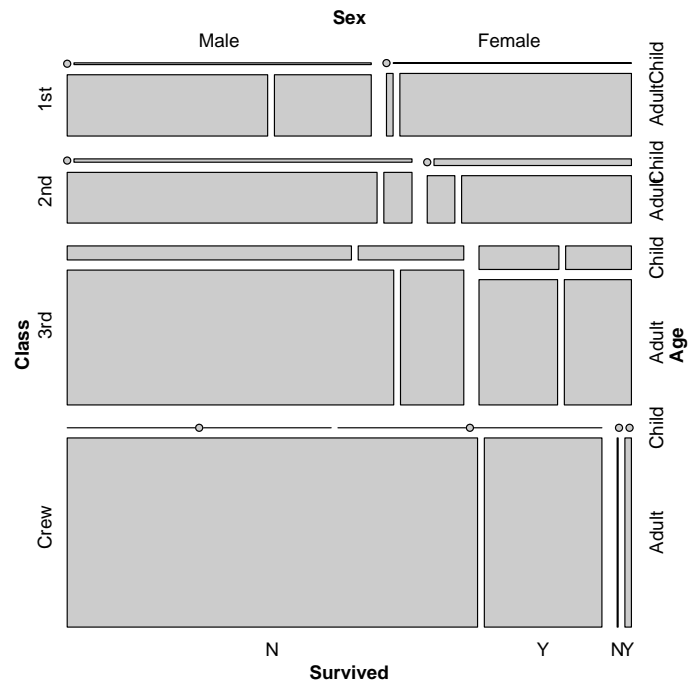


Figure 3: Abbreviating.

3 A simple list of labels: `labels_list()`

If problems with overlapping labels cannot satisfactorily resolved, the last remedy could be to simply list the levels below the plot (see Figure 14):

```
> mosaic(Titanic, labeling = labeling_list, margins = c(bottom = 5))
```

The number of columns can be specified.

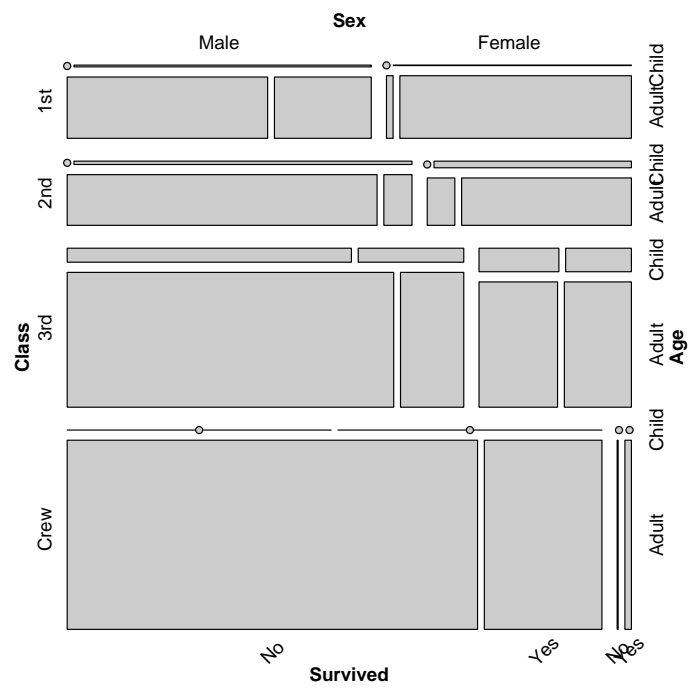


Figure 4: Rotating labels.

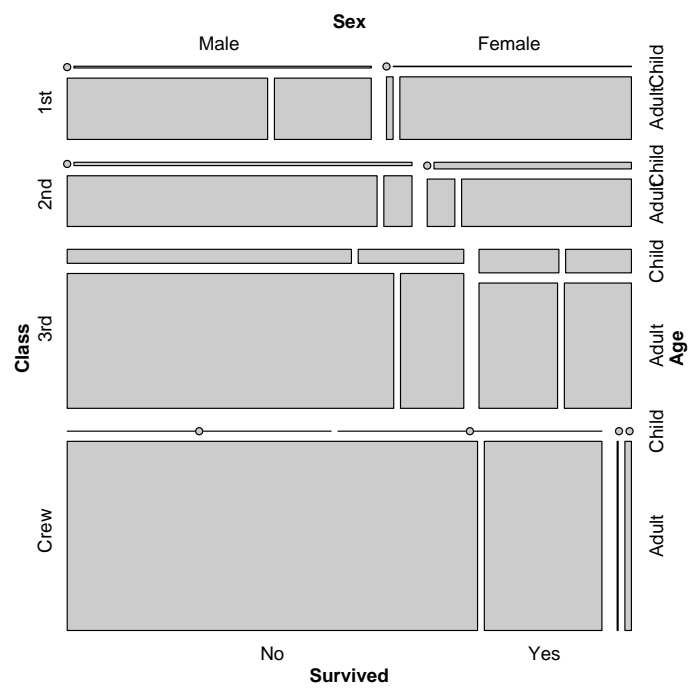


Figure 5: Inhibiting the repetition of levels.

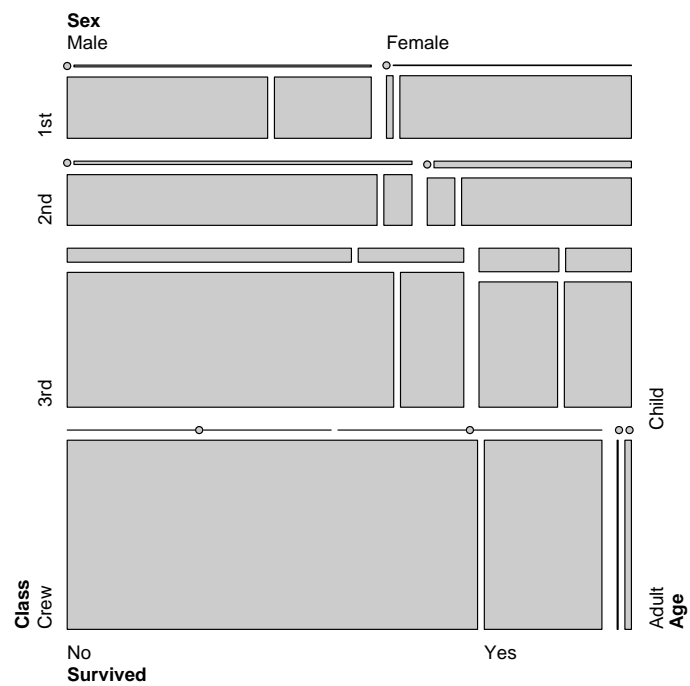


Figure 6: Left-aligning.

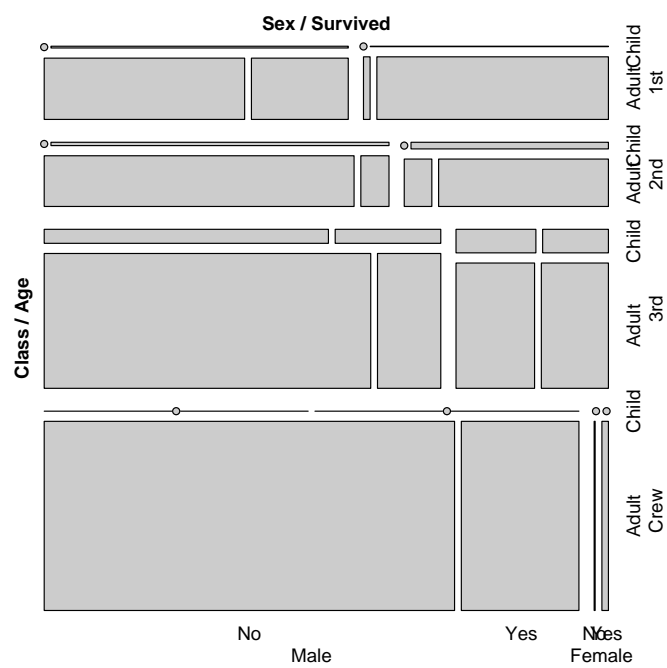


Figure 7: Changes in the margins.

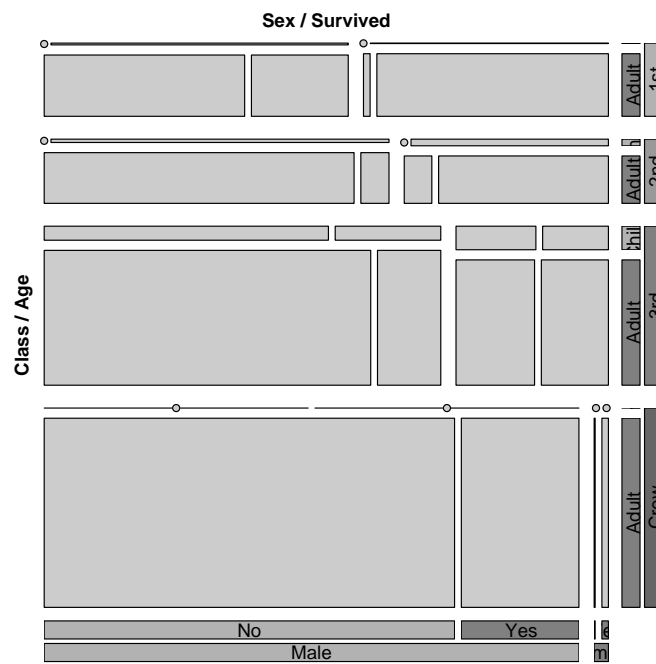


Figure 8: Boxes and Clipping.

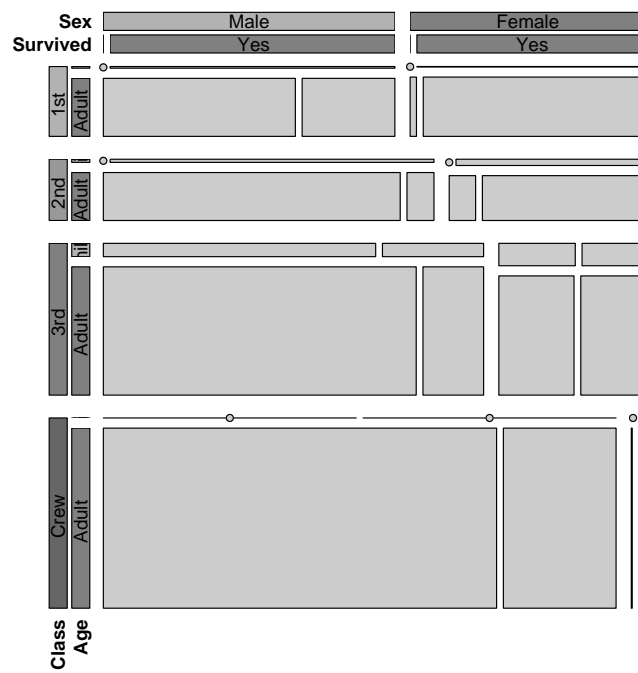


Figure 9: Variable names beneath levels.

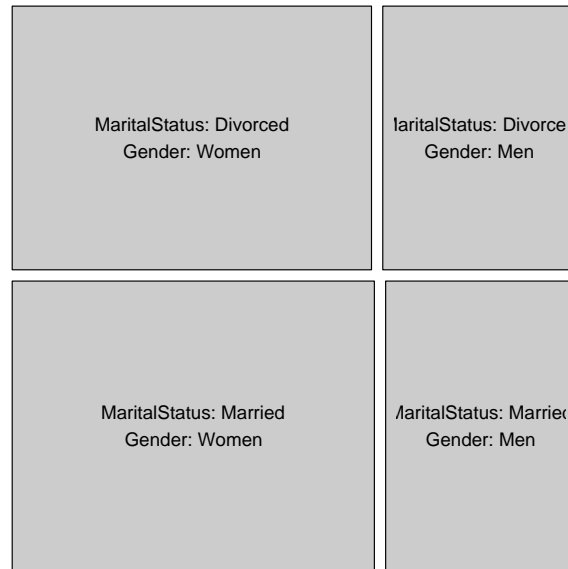


Figure 10: Cell labeling for the 'PreSex' data.

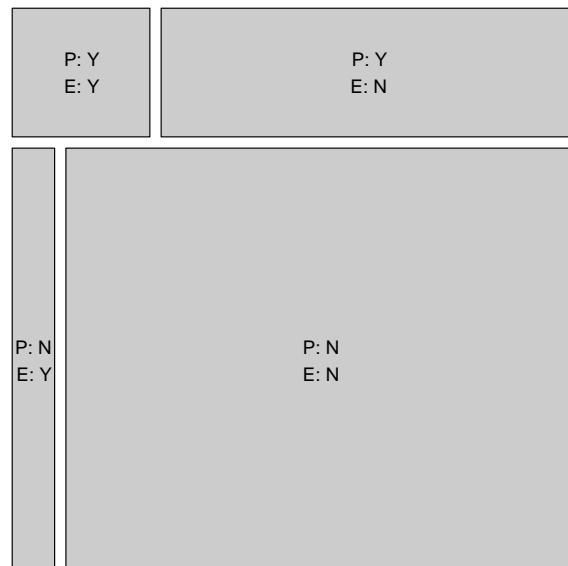


Figure 11: Cell labeling for the 'PreSex' data, labels abbreviated.

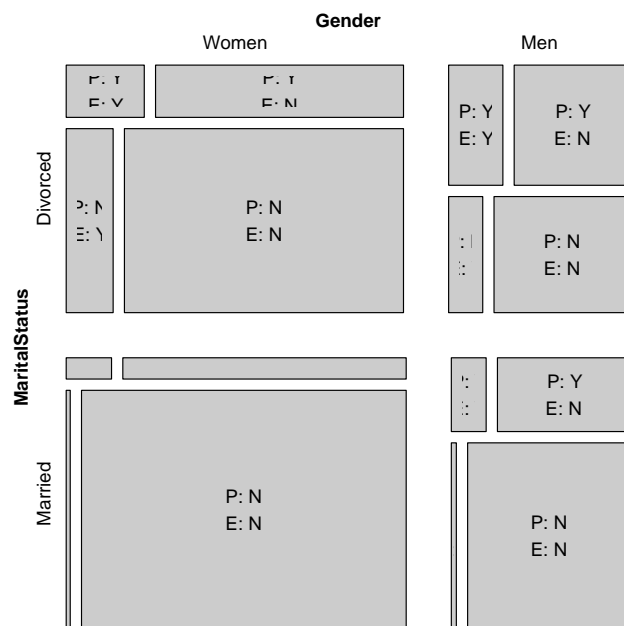


Figure 12: Conditional labeling for the 'PreSex' data, labels abbreviated.

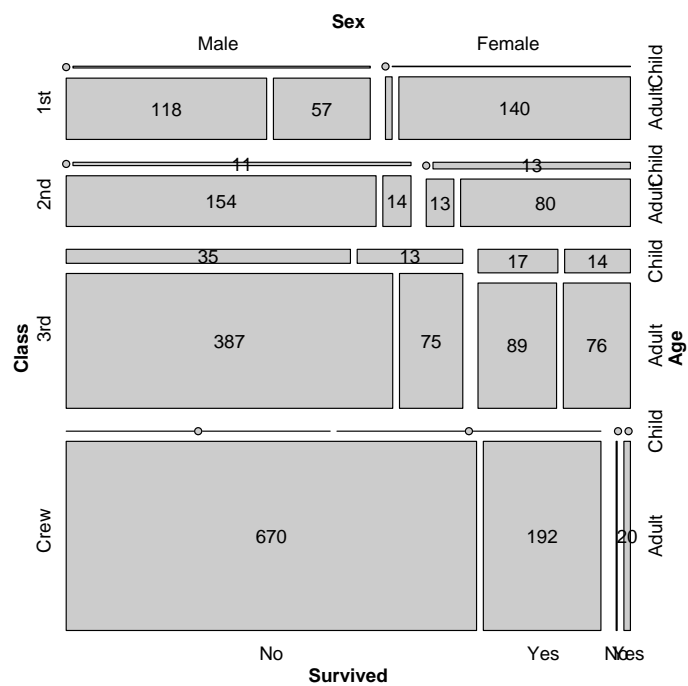


Figure 13: User-supplied Text added to a mosaic display of the 'Titanic' data.

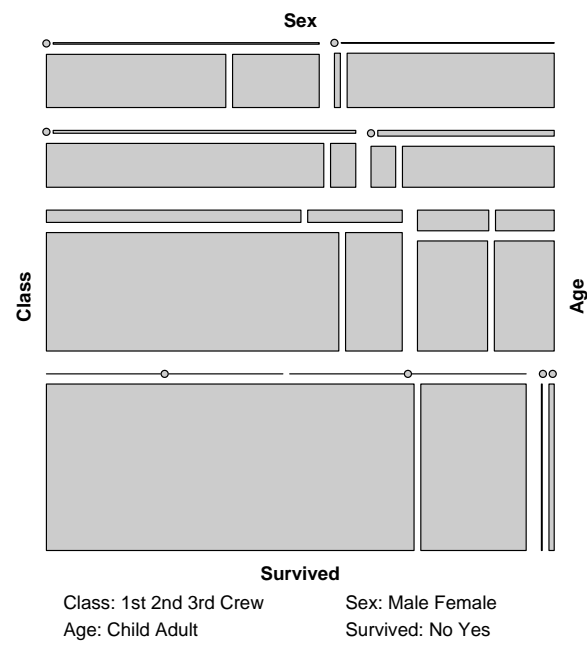


Figure 14: Labels indicated below the plot.