Labeling in the Strucplot Framework

by David Meyer, Achim Zeileis, and Kurt Hornik

1   Introduction

One of the major enhancements in package **vcd** compared to `mosaicplot()` and `assocplot()` in base R is the labeling in the strucplot framework which offers many more features and flexibility. Like shading, spacing, and drawing of legend and core plot, labeling is now carried out by specialized graphical appearance control ("grapcon") functions. This renders labeling completely modular (see the introductory vignette: "The Strucplot Framework—Visualizing Multi-way Contingency Tables" for an overview). 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 idea behind generating functions is discussed in detail in the vignette: "Colors and Residual-based Shadings in the Strucplot Framework"). Labeling is well-separated from the actual plotting that occurs in the low-level core functions. It only relies on the viewport tree produced by them, and the 'dimnames' attribute of the visualized table. Labeling functions are grapcons that "add ink to the canvas": the drawing of the labels happens after the actual plot has been drawn by the core function. 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. In the following, we describe the three basic modules (`labeling_text()`, `labeling_list()`, and `labeling_cells()`) and derived functions that build upon them.

2   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 in base R, but is much more flexible: it is not limited to 4 dimensions, and the positioning and graphical parameters of levels and variable names are customizable. In addition, the problem of overlapping labels can be handled in several ways.

As an example, consider the 'Titanic' data, consisting of 4 categorical variables: survival ('survived'), gender (i.e., '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, as well as some adult and child labels of the 'age' Variable. This issue can be addressed in several ways. The 'brute force' method is to enable clipping for these dimensions (see Figure 2):

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

The `clip` parameter is passed to the labeling function via the `labeling_args` argument which takes a list of parameters. `clip` itself takes a vector of logicals (one for each dimension). Almost all vectorized arguments in the strucplot framework can be abbreviated in the following way: unnamed components (or the defaults, if there are none) are recycled as needed, but overridden by the named components. Here, the default is `FALSE`, and therefore clipping is enabled only for the 'survived' and 'age' variables. A more sensible solution to the overlap problem is to abbreviate the levels (see Figure 3):
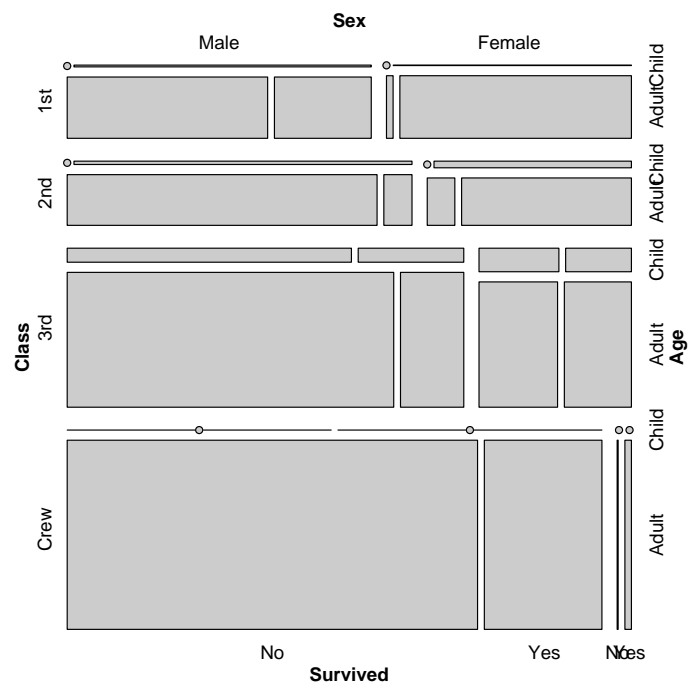
Figure 1: Mosaic plot for the 'Titanic' data with default settings for labeling.
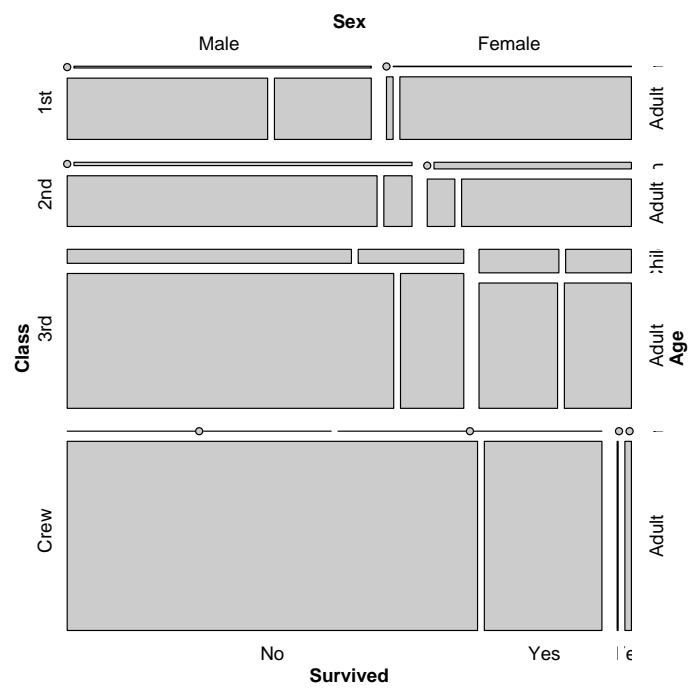

Figure 2: The effect of clipping.

```
> mosaic(Titanic, labeling_args = list(abbreviate = c(Survived = TRUE,
+     Age = 3)))
```
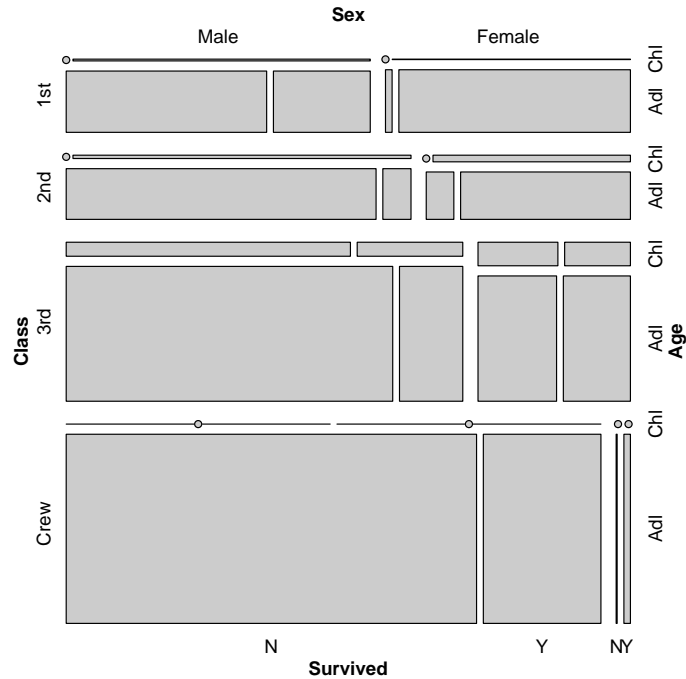

Figure 3: Abbreviating.

The abbreviate argument takes a vector of integers indicating the number of significant characters the levels should be abbreviated to (TRUE is interpreted as 1, obviously). Abbreviation is performed using the abbreviate() function in base R. Another possibility is to rotate the levels (see Figure 4):

```
> mosaic(Titanic, labeling_args = list(rot_labels = c(bottom = 90,
+     right = 0), offset_varnames = c(right = 1), offset_labels = c(right = 0.3)),
+     margins = c(right = 4, bottom = 3))
```

Finally, we could also inhibit the output of repeated levels (see Figure 5):

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

We now proceed with a few more 'cosmetic' features (which do not all produce satisfactory results for our sample data). A first simple, but effectful modification is to position all labels and variables left-aligned: (see Figure 6):

```
> mosaic(Titanic, labeling_args = list(pos_varnames = "left",
+     pos_labels = "left", just_labels = "left", rep = FALSE))
```

Note that obviously we need to change the justification to "left" as well. We can achieve the same effect by using the convenience function labeling_left():

```
> mosaic(Titanic, labeling = labeling_left)
```

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):
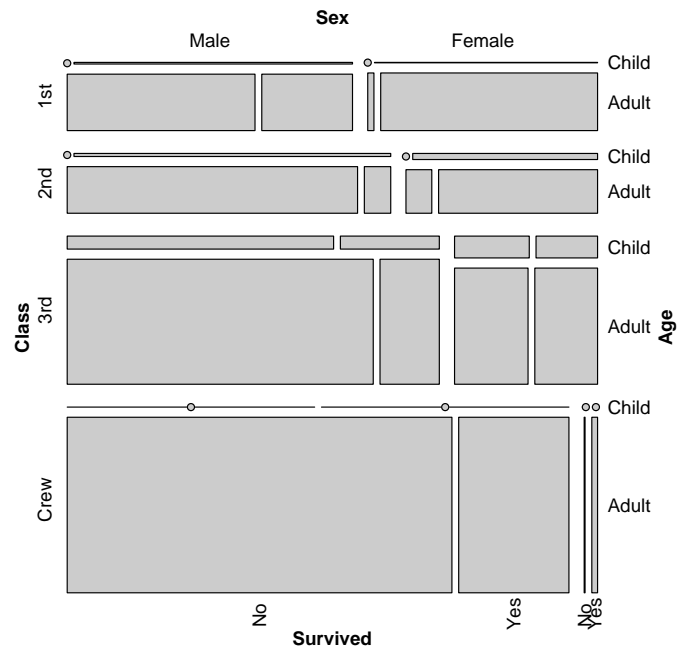
Figure 4: Rotating labels.

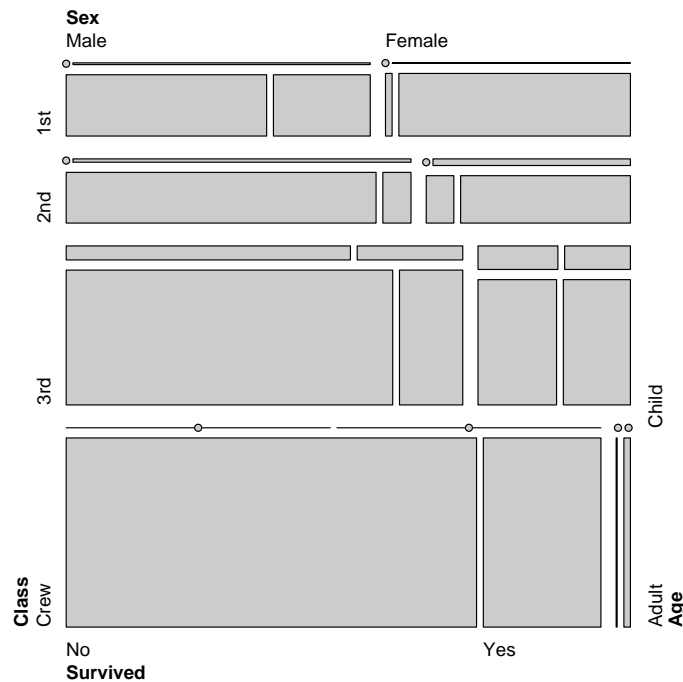Figure 5: Inhibiting the repetition of levels.

Figure 6: Left-aligning.

```
> mosaic(Titanic, labeling_args = list(tl_labels = FALSE,
+     tl_varnames = TRUE, abbreviate = c(Survived = 1,
+         Age = 3)))
```

The `tl_foo` ("top left") arguments are TRUE by default. 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 = TRUE))
```

The values to `boxes` and `clip` are recycled for all dimensions. The result is pretty close to what calling `mosaic()` with the `labeling_cboxed()` wrapper does, except that variables and levels, by default, are put to the top and to the left of the plot:

```
> mosaic(Titanic, labeling = labeling_cboxed)
```

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 = c(Survived = FALSE, TRUE), labbl_varnames = TRUE),
+     margins = c(left = 4, right = 1, 3))
```

`labbl_varnames` ("variable names to the bottom/left of the labels") is a vector of logicals indicating the side for the variable names. The resulting layout is close to what `labeling_lboxed()` produces, except that variables and levels, by default, are left-aligned and put to the bottom and to the right of the plot:

```
> mosaic(Titanic, labeling = labeling_lboxed, margins = c(right = 4,
+     left = 1, 3))
```

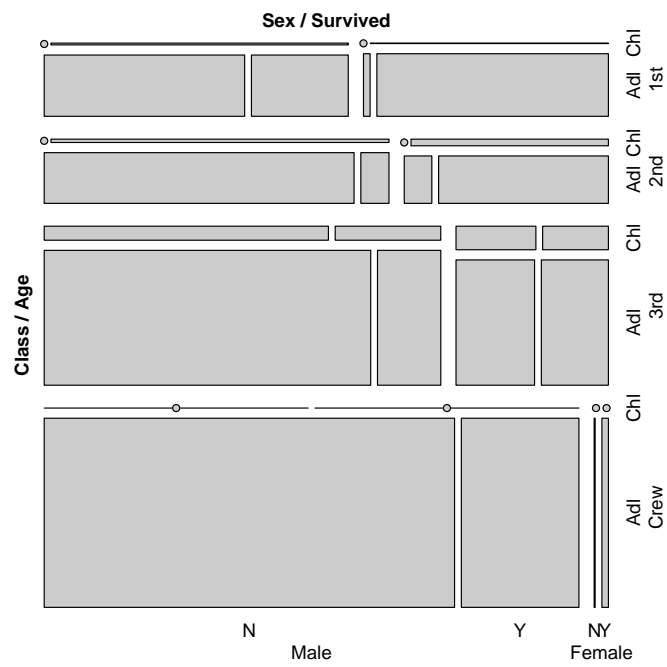A similar design is used by the `doubledecker()` function.
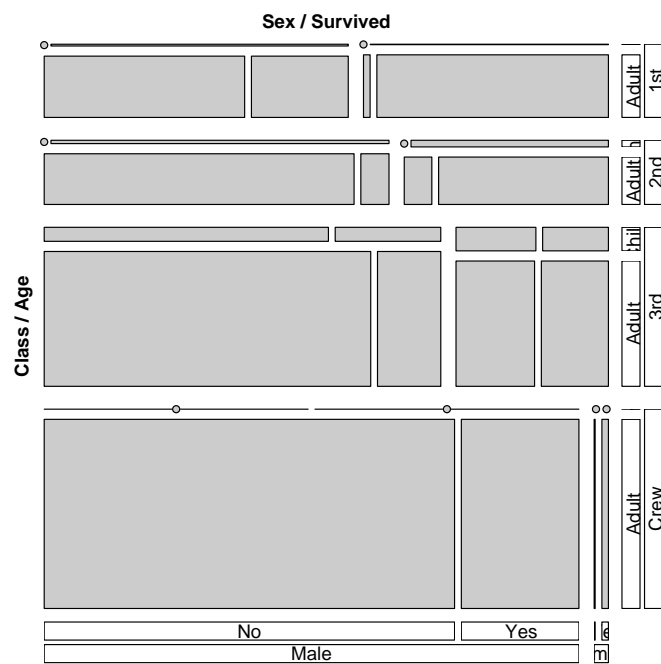
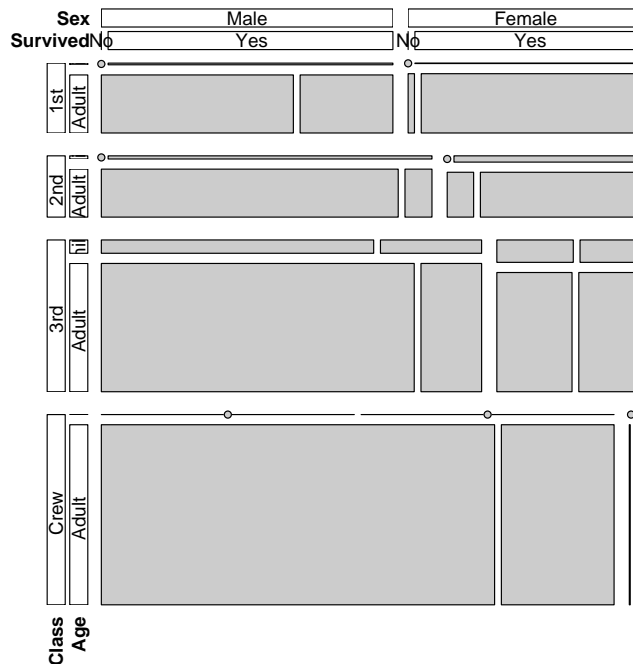Figure 7: Changes in the margins.

Figure 8: Boxes and Clipping.

Figure 9: Variable names beneath levels.

3 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, it might be useful to abbreviate labels and/or variable names and turn off clipping (see Figure 11):

```
> mosaic(~PremaritalSex + ExtramaritalSex, data = PreSex,
+     labeling = labeling_cells(abbreviate_labels = TRUE,
+         abbreviate_varnames = TRUE, clip = FALSE))
```

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 = labeling_conditional(abbreviate_varnames = TRUE,
+     abbreviate_labels = TRUE, clip = FALSE))
```

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 using the 'Titanic' data, this is used to add all observed values greater than 5 to the cells after the mosaic has been plotted (see Figure 13):

```
> mosaic(Titanic, labeling_args = list(abbreviate = c(Survived = 1,
+     Age = 4)), pop = FALSE)
> tab <- ifelse(Titanic < 6, NA, Titanic)
> labeling_cells(text = tab, clip = FALSE)(Titanic)
```

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

4   A simple list of labels: `labeling_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))
```

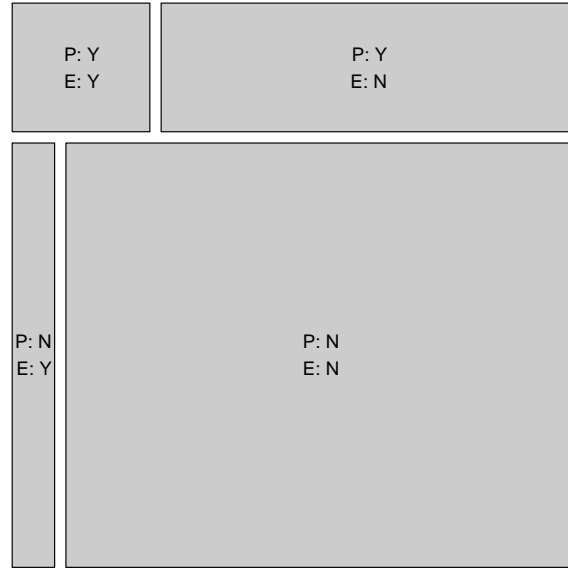The number of columns can be specified.

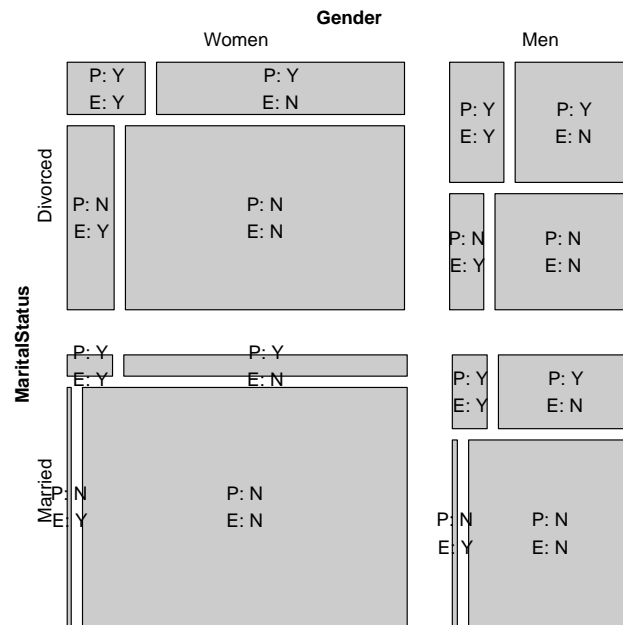Figure 11: Cell labeling for the 'PreSex' data, labels abbreviated.

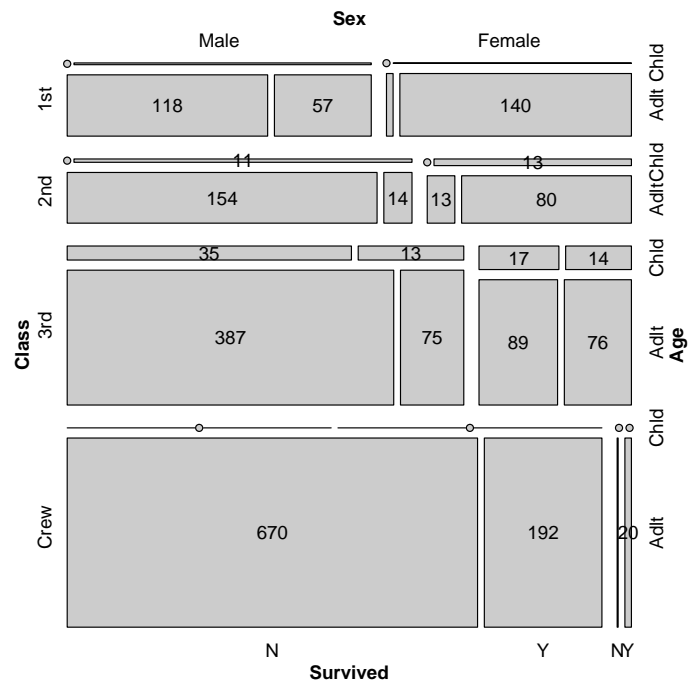Figure 12: Conditional labeling for the 'PreSex', labels abbreviated.

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

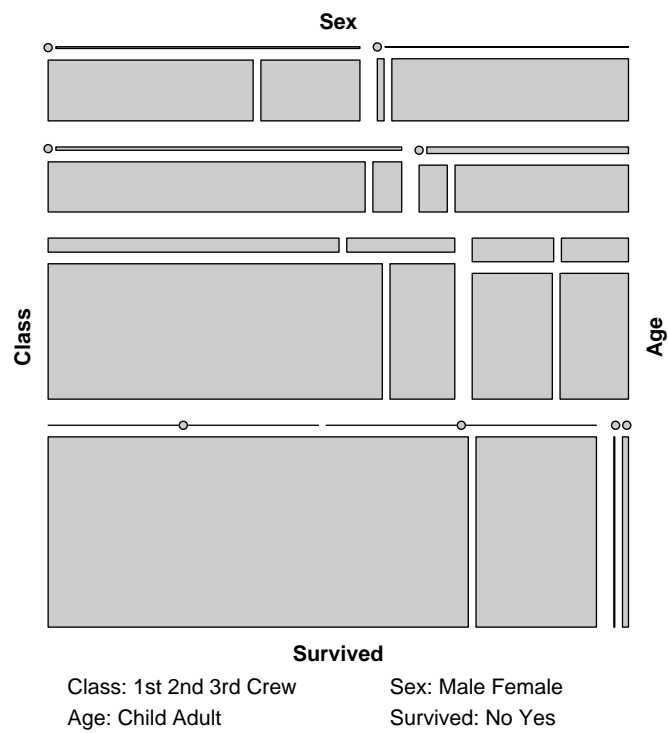
Class: 1st 2nd 3rd Crew   Sex: Male Female
Age: Child Adult   Survived: No Yes

Figure 14: Labels indicated below the plot.