Colors and residual-based Shadings in Strucplots

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

August 26, 2005

Unlike other R graphics functions, strucplots allow almost full control over the graphical parameters of all plot elements. In particular, the user can *independently* modify the appearance of the tiles of association plots and mosaic plots. Built on top of this functionality, the framework supplies a set of shading functions that choose colors appropriate to the visualization of independence models fit to the visualized data. The tiles' graphical parameters are set using the `gp` argument of the strucplot functions. This argument basically expects an object of class `gpar` whose components have the same shape (length and dimensionality) than the data table (see Section 1). For convenience, however, the user can also supply a function that computes such an object given a vector of residuals, or, alternatively, a generating function that takes parameters and returns such a function (see Section 2).

1 Specifying graphical parameters in strucplots

As an example, consider the 'Admissions to the University of California in Berkely (UCB)' data. In the table collapsed over departments, we would like to highlight the (incidentally wrong) impression that there were to much male students accepted compared to the presumably discriminated female students (see Figure 1):

```
> (ucb <- margin.table(UCBAdmissions, 1:2))
```

```
          Gender
Admit      Male Female
  Admitted 1198    557
  Rejected 1493   1278
```

```
> (fill_colors <- matrix(c("dark cyan", "gray", "gray", "dark magenta"),
+     ncol = 2))
```

```
     [,1]        [,2]
[1,] "dark cyan" "gray"
[2,] "gray"      "dark magenta"
```

```
> mosaic(ucb, gp = gpar(fill = fill_colors, col = 0))
```

As the example shows, we therefore create a fourfold table with appropriate colors and supply them to the `fill` component of the `gpar` object passed to the `gp` argument of `mosaic()`. In addition, we use the `col` component to turn off the borders for cosmetic reasons.

If the parameters specified in the `gpar` object are incomplete, strucplot functions will recycle them along the last splitting dimension. In the following example based on the 'Survival on the Titanic' data, we will highlight all cells corresponding to survived passengers (see Figure 2):

```
> mosaic(Titanic, gp = gpar(fill = c("gray", "dark magenta")))
```

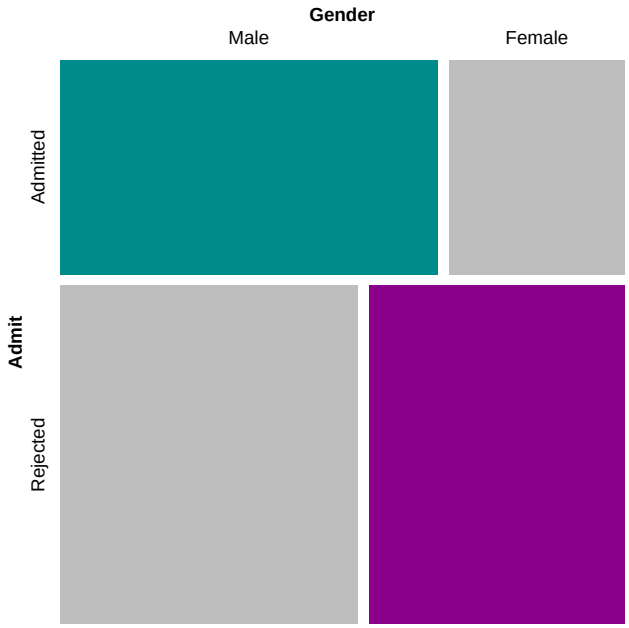

Figure 1: Mosaic plot for the UCBAdmissions data with highlighted cells.

2 Using residual-based shadings

This new flexible way of specifying graphical parameters is the basis for a suite of shading functions that modify the tiles' appearance with respect to a vector of residuals, resulting from deviations of the observed from the expected values under a given independence model. The idea is to visualize at least the size of the residuals, but some shadings, additionally, indicate overall significance. One particular shading, the maximum shading, even allows to identify those cells that cause the rejection of the independence hypothesis.

All shading functions are parameterized be the means of generating functions that return the actual shading functions. As an example, consider the `shading_binary()` function that only visualizes the sign of the residuals:

```
shading_binary <- function(observed = NULL, residuals = NULL, expected = NULL,
                           df = NULL, col = hcl(c(260, 0), 50, 70))
{
\dots
}
```

Each generating function, passed to a strucplot functions' `gp` argument, takes a few arguments that describe the model (`observed`, `expected`, `residuals`, and `df`), and a number of parameters that specify the appearance (in the case of `shading_binary()` only `col`, a vector of length two, that specifies the two colors corresponding to negative and positive residuals, respectively). The generating function then return the actual shading function taking only one argument: a vector of residuals, and returning an object of class `gpar` that contains the graphical parameters for all tiles corresponding to the residuals. In the following example, positive and negative residuals of an hypothesized independence model shall be visualized by different colors (see Figure 3):

```
> obs <- margin.table(UCBAdmissions, 1:2)
> exp <- independence_table(obs)
> (res <- (obs - exp)/sqrt(exp))
```

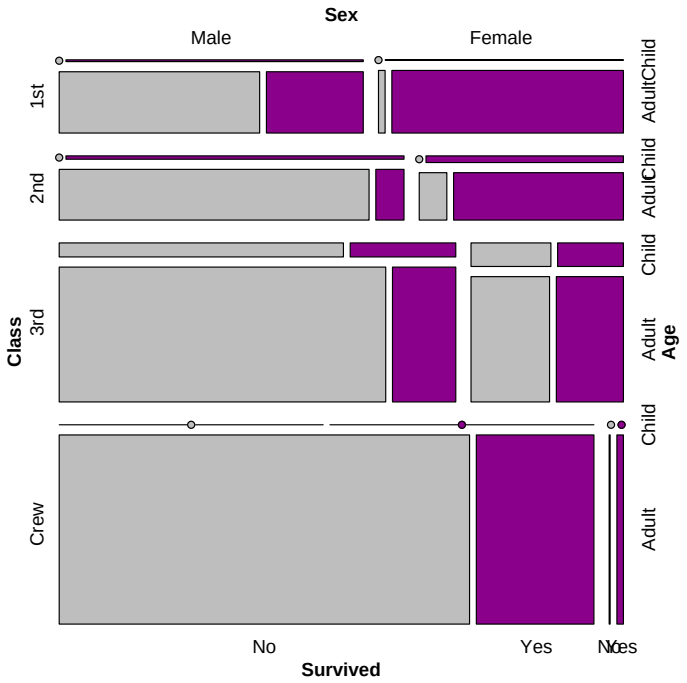

Figure 2: Recycling of parameters.

```
          Gender
Admit           Male    Female
  Admitted  4.784093 -5.793466
  Rejected -3.807325  4.610614
```

```
> (gpobj <- shading_binary()(res))
```

```
$fill
     [,1]      [,2]
[1,] "#96A2DF" "#E18E9E"
[2,] "#E18E9E" "#96A2DF"
```

```
> mosaic(obs, gp = gpobj)
```

The shading-generating functions available in **vcd** include model-fitting, and hence the R code corresponding to the example above in fact reduces to:

```
> mosaic(obs, gp = shading_binary)
```

3 An overview of the shading functions in vcd

Currently, there are three basic shadings available (`shading_hcl()`, `shading_hsv()`, and `shading_binary()`, as well as two derived functions (`shading_Friendly()` building upon `shading_hsv()` and `shading_max()` building upon `shading_hcl`). `shading_binary()` has been explained in the previous paragraph. `shading_hsv()` and `shading_hcl()` are quite similar in their functionality, but use different color spaces: the Hue-Saturation-Value (HSV) and the Hue-Chroma-Luminance (HCL) scheme, respectively. We will focus on the latter, since the HCL space is preferable to the HSV space in many respects (e.g., uniform saturations over different hues, device independentness, ...).

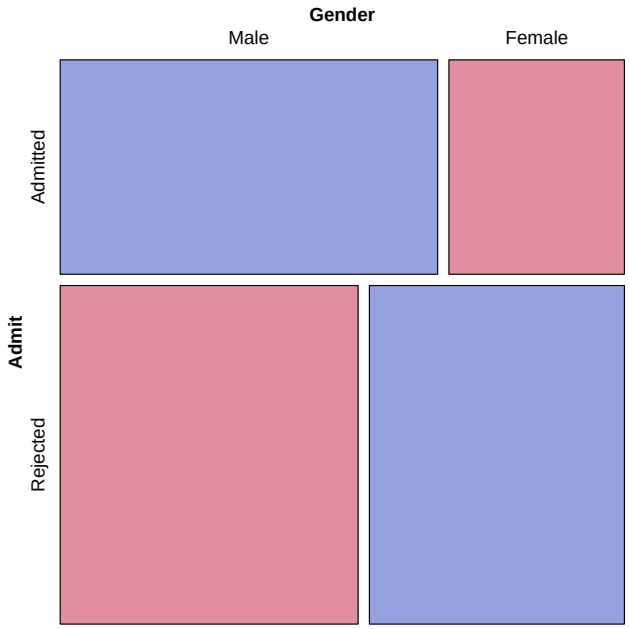

Figure 3: Binary shading.

In the HCL space, colors are specified in three dimensions: hue, chroma ('colorfulness', saturation), and luminance ('brightness', amount of gray). These three dimensions are used by `shading_hcl()` to visualize information about the residuals and the underlying independence model. A residuals' hue indicates its sign (by default blue and red for positive and negative ones, respectively). An overall full or poor chroma indicates significant or non-significant results, respectively. Finally, the luminance of a residual is set according to its size.

As an example, we will visualize the (in)dependence of hair and eye color in the 'Hair and Eye Color' data set (see Figure 4)

```
> haireye <- margin.table(HairEyeColor, 1:2)
> mosaic(haireye, shade = TRUE)
```

The default shading scheme is `shading_hcl()`, so it suffices to turn shading on to bring colors to the mosaic plot. Large positive residuals (greater than 4) can be found for brown eyes/black hair and blue eyes/blond hair, they are colored in saturated red. On the other side, there is a large negative residual (less than -4) for brown eyes/blond hair, colored deep blue. There are also three medium-sized positive (negative) residuals between 2 and 4 (-2 and -4): the colors for them are less saturated. The heuristic for choosing the cut-off points 2 and 4 is that the Pearson residuals are approximately standard normal which implies that the highlighted cells are those with residuals *individually* significant at approximately the $\alpha = 0.05$ and $\alpha = 0.0001$ level. These default cut-off points can be changed to alternative values using the `interpolate` argument (see Figure 5):

```
> mosaic(haireye, shade = TRUE, gp_args = list(interpolate = 1:4, h = c(100,
+     200)))
```

Note that we also changed the hue values for purely cosmetic reasons. The third remaining dimension, the luminance, is used for visualizing the significance of a test statistic. The p value can be supplied to the `p.value` argument. If none is specified, it is computed from a Chi-squared

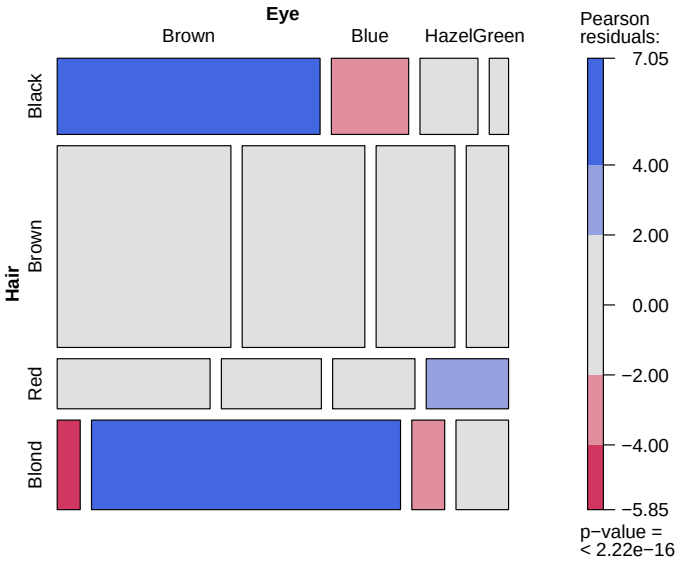

Figure 4: Shaded residuals in the HairEyeColor data set—two cut-off points

distribution with `df` degrees of freedom. The `level` argument is used to specify the confidence level: if `p.value` is smaller than `1 - level`, bright colors are used, otherwise dark colors are employed. These colors are computed using the two luminance values supplied to the `l` argument. The following example using the 'Bundesliga' data shows the relationship of home goals and away goals of the German football league in 1995: although there are two "larger" residuals (one greater than 2, one less then −2), the Chi-squared test does not reject the null hypothesis of independence. Consequently, the colors appear dark (see Figure 6):

```
> bl <- xtabs(~HomeGoals + AwayGoals, data = Bundesliga, subset = Year ==
+     1995)
> mosaic(bl, shade = TRUE)
```

The `interpolate` argument also accepts a function that automatically chooses the cut-off points in a data-driven way. This functionality is used by `shading_max()` that uses the maximum statistic to visualize significant *cells* causing the rejection of the independence hypothesis. The `level` argument of `shading_max()` then can be used to specify several confidence levels from which the corresponding cut-off points are computed. By default, two cut-off points are computed corresponding to confidence levels of 90% and 99%, respectively. In the following example, we investigate the effect of a new treatment for rheumatoid arthritis on a group of female patients using the maximum shading (see Figure 7):

```
> art <- xtabs(~Improved + Treatment, data = Arthritis, subset = Sex ==
+     "Female")
> mosaic(art, gp = shading_max)
```

The maximum test is significant although the residuals are all in the $[-2, 2]$ interval, as the cut-off points are even smaller. The `shading_hcl` function with default cut-off points would not have shown any color. In addition, since the test statistic is the maximum of the absolute pearson residuals, *every* colored residual violates the hypotheses of independence, and thus, the 'culprits' can immediately be identified.

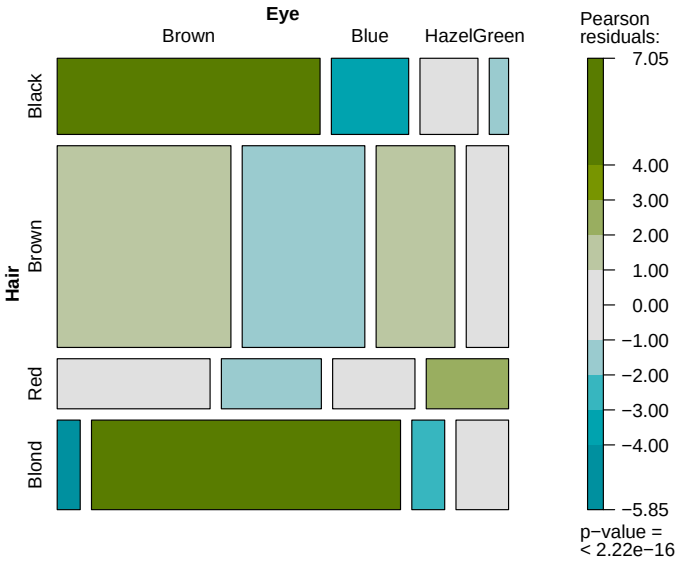

Figure 5: Shaded residuals in the HairEyeColor data set—four cut-off points

The last shading function is `shading_Friendly()`. This shading mimics the shading introduced by Friendly (2000). This shading corresponds to the HSV shading with default cut-off points (2 and 4), and in addition uses the border color and line type to visualize the residuals' sign. The following example again visualizes the Bundesliga data from above, this time using the Friendly scheme and, in addition, an alternative legend:

```
> mosaic(bl, gp = shading_Friendly, legend = legend_fixed, zero_size = 0)
```

(The bullets indicating zero observed values are removed here since this feature does not exist in the original SAS implementation of the Friendly mosaic plots).

4 Creating new shading functions

The example with four cut-off points (Figure 5) might suggest that a continuous shading simply visualizing the complete range of confidence levels might be useful. The following code shows how such a shading function might be constructed:

```
> shading_continuous <- function(observed = NULL, residuals = NULL, expected = NULL,
+     df = NULL, n = 5000, ...) {
+     stopifnot(length(dim(observed)) == 2)
+     obs.test <- coindep_test(observed, n = n)
+     ipol <- function(x) abs(obs.test$pdist(x))
+     shading_hcl(observed = NULL, residuals = NULL, expected = NULL,
+         df = NULL, interpolate = ipol, p.value = obs.test$p.value, ...)
+ }
> class(shading_continuous) <- "panel_generator"
> mosaic(haireye, gp = shading_continuous)
```

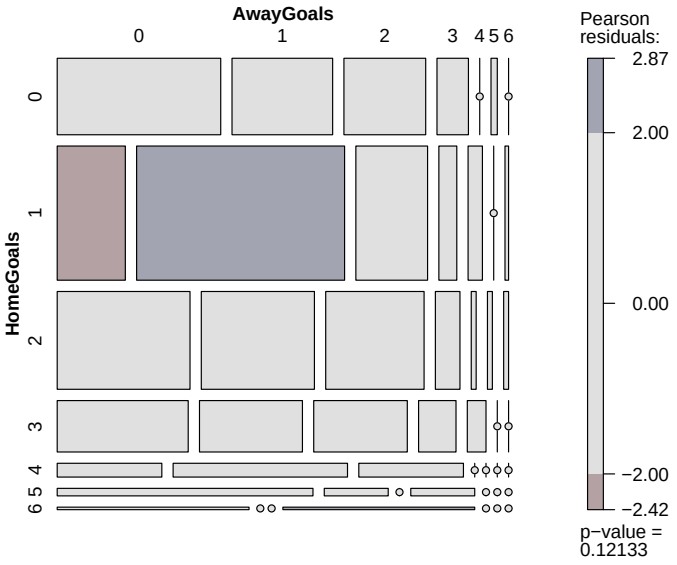

Figure 6: Non-significant Chi-squared test using part of the Bundesliga data.

We first compute the permutation distribution for the maximum statistic using `coindep_test`, and then simply use it as the interpolation function passed to `shading_hcl`. The result using the HairEyeColor data, however, is disappointing: too much color makes it difficult to interpret the image, and the subtle color differences are hard to catch. Therefore, we only included shadings with discrete cut-off points.

References

Friendly, M. (2000). *Visualizing Categorical Data*. SAS Insitute, Carey, NC.

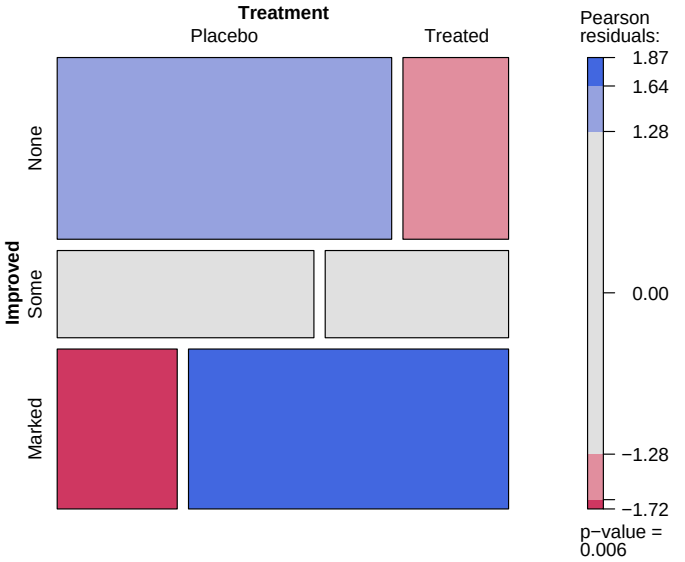

Figure 7: Significant maximum test on female patients of the Arthritis data.

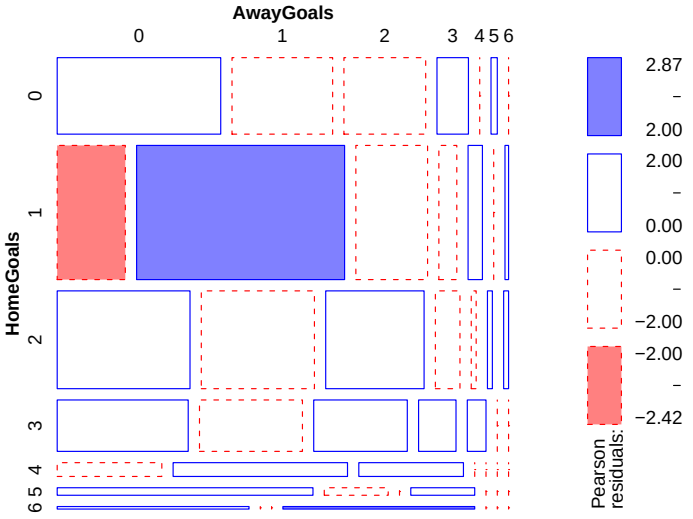

Figure 8: The Bundesliga data for 1995 using the Friendly shading.

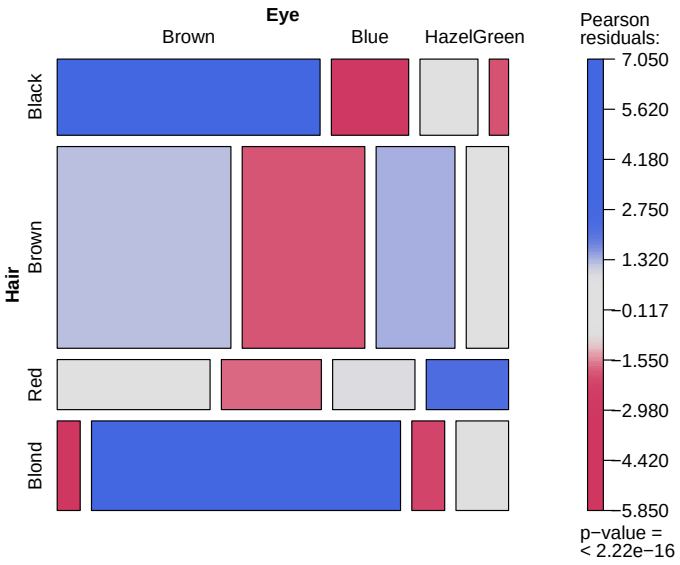

Figure 9: The HairEyeColor data with continuous shading.