University of Bern Social Sciences Working Paper No. 31

Color palettes for Stata graphics

Ben Jann

April 12, 2018

http://ideas.repec.org/p/bss/wpaper/31.html
http://econpapers.repec.org/paper/bsswpaper/31.htm
Color palettes for Stata graphics

Ben Jann
University of Bern

Abstract. This paper introduces a command called colorpalette that provides a wide variety of color palettes and color generators for use in Stata graphics. Supported are color palettes from official Stata’s graph schemes, a selection of palettes that have been proposed by users, standard collections such as the ColorBrewer or D3.js palettes, as well as HSV and HCL color generators. As a byproduct, the paper also introduces commands for marker symbol and line pattern palettes.

Keywords: st0001, palettes, colorpalette, symbolpalette, linepalette, graph, graphics, color, color spaces

1 Introduction

Stata features a set of about 50 named colors that can be used in graphs (see [G] colorstyle). Given the diverse needs of users, a set of 50 predefined colors is rather limited. Alternative colors are supported, but have to be specified by their RGB, CMYK, or HSV values. To increase the number of easily accessible colors, the new colorpalette command provides a wide variety of predefined palettes and also features HSV (Hue-Saturation-Value) and HCL (Hue-Chroma-Luminance) color generators. Furthermore, it supports additional input formats for custom colors, such as hex triplets, and allows generating colors over a range if intensity or opacity levels.

A byproduct of colorpalette are two additional commands, symbolpalette and linepalette, that provide palettes of marker symbols and line pattern. These commands are briefly presented in the Appendix.

To install the new commands and view the documentation, type:

.scc install palettes, replace
.help palettes

2 Syntax and basic usage

The colorpalette command has two syntax variants. Syntax 1 is used to retrieve colors from one or multiple palettes. The colors are returned in r() and, by default, displayed in a graph. The syntax is

colorpalette [argument] [ , palette_options graph_options ]

where argument is
Color palettes for Stata graphics

```stata
palette [ , palette_options] / [ palette [, palette_options] / ... ]
```

and `palette` is a named palette as described below, or a space-separated list of named colors, RGB values, CMYK values, or HSV values according to `[g] colorstyle`. HCL values specified as "hcl h c l", where `h` specifies the hue (dominant wavelength in degrees of the 360 degree color wheel), `c` specifies the chroma (colorfulness; $c \geq 0$), and `l` specifies the luminance (brightness, amount of gray; $l \in [0, 100]$), or hex triplets specified as `#rrggbb`, where `rr`, `gg`, and `bb` are the two digit hex codes (or one digit abbreviations) for red, green, and blue. The specified colors can include intensity adjustment and, since Stata 15, an opacity level specified as "color */int/[%op]", where `int \in [0, 1)` makes the color lighter, `int > 1` makes the color darker, and `op` is a number between 0 (fully transparent) and 100 (fully opaque).

Syntax 2 is used to display an overview of multiple palettes in a single graph, without returning the colors in `r()`. The syntax is

```stata
colorpalette [ , palette_options graph_options]: pspec [ / pspec / ... ]
```

where `pspec` is

```stata
palette [ , palette_options]
```

or . to insert a gap.

Palette options

`n(#)` specifies the size of the palette (the number of colors). In many cases this just selects the first `#` colors from the palette and is thus equivalent to `select(1/#)`. However, some color schemes return colors that adjust to the size of the palette.

`select(numlist)` selects and orders the colors retrieved from the palette.

`reverse` returns the palette in reverse order.

`intensity(numlist)` applies color intensity adjustment. Values between 0 and 1 make the colors lighter; values larger than one make the colors darker. Specify multiple values to use different adjustments across the selected colors. The list of adjustments will be recycled if it is shorter than the list of selected colors. Likewise, colors will be recycled if the list of adjustments is longer than the palette.

`opacity(numlist)` sets the opacity level(s) (requires Stata 15). Values must be between 0 (fully transparent) and 100 (fully opaque). Specify multiple values to use different opacity levels across the selected colors. The list of opacity levels will be recycled if it is shorter than the list of selected colors. Likewise, colors will be recycled if the list of opacity levels is longer than the palette.

The above options are supported by all palettes, but some palettes also have additional options; see the descriptions of the palettes below.
Common graph options

*title*(string) specifies a custom title for the graph.

*gropts*(twoway_options) provides options to be passed through to the graph command; see [G] `twoway_options`.

Additional graph options for syntax 1

*nograph* suppresses the graph.

*rows(#)* specifies the minimum number of rows in the graph. The default is 5.

Additional graph options for syntax 2

*horizontal* displays the palettes horizontally. This is the default.

*vertical* displays the palettes vertically.

*plabels*(strlist) provides custom labels for the palettes. Enclose labels with spaces in double quotes.

*color*(colorstyle) specifies a custom outline color. The default is to use the same color as for the fill.

*lwidth*(linewidthstyle) specifies a custom outline thickness. The default is `lw(vthin)`.

Stored results

Under syntax 1, *colorpalette* stores the following in r():

Scalars:

- r(n) number of colors

Macros:

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r(ptype)</td>
<td>color palette name</td>
</tr>
<tr>
<td>r(pnote)</td>
<td>palette note (if available)</td>
</tr>
<tr>
<td>r(p#)</td>
<td>#th color</td>
</tr>
<tr>
<td>r(p#info)</td>
<td>info on #th color (if available)</td>
</tr>
</tbody>
</table>

2.1 Viewing a palette (syntax 1)

To display a single palette, type *colorpalette* followed by the name of the palette. For example, to view the *economist* palette, type:
The graph produced by colorpalette displays the colors as well as their names or color codes and, possibly, some additional information. Here is an example of a semantic palette by Lin et al. (2013) with RGB codes and labels:

```
. colorpalette economist
```

It is also possible to combine colors from multiple palettes. Here is an example that uses the fruits palette, the vegetable palette, and a selection of colors from the food palette by Lin et al. (2013):
. colorpalette lin, fruits
>    / lin, vegetable
>    / lin, food select(1/6)
>    title("Edibles")

2.2 Retrieving colors from a palette (syntax 1)

`colorpalette` returns the values of the colors in r() so that they can be used in a
subsequent graph command. r(p) will contain a space separated list of all colors,
r(p1), r(p2), etc. will contain the single colors one by one. Here is an example that
selects four colors from ColorBrewer’s `Set1` palette and uses them in a line plot (option
`nograph` is specified to prevent `colorpalette` from displaying the palette):

```
. sysuse uslifexp, clear
(U.S. life expectancy, 1900-1999)
. lab var le_wfemale "white females"
. lab var le_umale "white males"
. lab var le_bfemale "black females"
. lab var le_bmale "black males"
. colorpalette Set1, select(1/3 5)
>    nograph
. line le_wfemale le_umale
>    le_bfemale le_bmale year
>    , lcolor(‘r(p)’) lwidth(*2 .)
>    ytitle(Life expectancy)
```

Macro expansion notation `r(p)` instructs Stata to paste the contents of r(p) at
the specified position within the command. Note that many commands, including most
graph commands, clear r(). That is, if you want to use the same colors in multiple
graphs without having to call `colorpalette` repeatedly, copy the colors to a local or
global macro ([P] `macro`). For example, typing

```
. local mycolors "'r(p)'"'
```
would copy the list of colors to local macro `mycolor`. You could then use the colors in subsequent graph commands by typing `mycolors`.

An alternative is to use the `grstyle` command to change the default colors used in Stata graphs; `grstyle` calls `colorpalette` internally (see Jann 2018).

### 2.3 Viewing multiple palettes (syntax 2)

To display an overview of multiple palettes in a single graph, type `colorpalette`, a colon, and a list of palettes separated by forward slashes. The following example displays some of the categorical palettes from ColorBrewer (Brewer 2016; Brewer et al. 2003):

```
.colorpalette, lcolor(black):
> Accent / Dark2 / Set1 / Set3
```

Option `lcolor(black)` has been specified to draw black lines around the color fields. Separate options can be specified for each palette. Here is an example that displays several default schemes of the HCL color generator. The example also illustrates the effect of the `vertical` option and the use of `n()` to determine the number of colors:

```
.colorpalette, vertical n(40):
> hcl, blues / hcl, greens /
> hcl, oranges / hcl, purples /
> hcl, heat / hcl, plasma
```
### 2.4 Specifying a custom list of colors

Instead of using a named palette you can provide a custom palette by specifying a list of `colorstyles` (named colors, RBG values, CMYK values, or HSV values; see [G `colorstyle`]). Here is an example displaying some of Stata’s named colors:

```stata
.colorpalette blue brown cranberry
  >   emerald forest_green gold green
  >   khaki lavender lime magenta
  >   maroon mint navy olive
  >   olive_teal orange orange_red
  >   pink purple red sand sienna
  >   teal, title(Some named colors)
```

In addition to the color specifications documented in [G `colorstyle`], you can specify colors using HCL codes (type `"hcl h c l"`, where `h`, `c`, and `l` are the values for hue, chroma, and luminance) or hex triplets. The specified colors will be translated to RGB. Here is an example displaying (approximate) Federal Standard 595 Safety Colors, using the hex codes found at www.w3schools.com/colors/colors_fs595.asp:

```stata
.colorpalette #bd1e24 #e97600
  >   #f6c700 #007256 #0067a7
  >   #964f8e, rows(2)
  >   title(FS 595 Safety Colors)
```

### 2.5 Creating colors over a range of intensities or opacity levels

The `intensity()` and `opacity()` options can be used to apply intensity adjustment or assign opacity levels to the selected colors. Both options support number lists as
argument (see [U] 11.1.8 numlist). If the list of specified numbers is longer than the number of colors in the palette, the list of colors will be recycled. This allows creating colors over a range of intensities or opacity levels, as in the following example:

```
colorpalette cranberry, intensity(0.1(.05)1)
```

```
1: cranberry*.1 8: cranberry*.45 15: cranberry*.8
7: cranberry*.4 14: cranberry*.75
```

### 2.6 Custom palettes

If you want to create an own named color palette, you can define a program called `colorpalette_myname`, where `myname` is the name of your palette. Your program should return the color definitions (e.g., RGB values or hex codes) as a comma-separated list in local macro `P`. In addition you may provide a comma-separated list of descriptions in local macro `I`.

After defining the program, the new palette is available to `colorpalette` like any other palette. Here is an example providing a palette called `bootstrap3` containing semantic colors used for buttons in Bootstrap v3.3 (getbootstrap.com/docs/3.3):

```
program colorpalette_bootstrap3
1. c_local P #337ab7,#5cb85c,#5bc0de,#f0ad4e,#d9534f
   > primary,success,info,warning,danger
2. c_local I primary,success,info,warning,danger
3. end
.colorpalette bootstrap3
```

```
1: 51 122 183 primary
2: 92 184 92 success
3: 91 192 222 info
4: 240 173 78 warning
5: 217 83 79 danger
```
More complicated definitions of palettes that take account of specific options are also possible. See the palette definitions in colorpalette.ado for examples.

If you intend to use the new palette in different analyses, you can store the program in an ado-file instead of including it in each of the do-files. If myname is the name of your palette, the program should be stored in file colorpalette_myname.ado in the working directory or somewhere along Stata’s ado path (see [P] sysdir).

3 Predefined color palettes

This section provides an overview of the named palettes implemented in colorpalette. There are three types of palettes: Palettes providing the colors used for plots 1 to 15 in official Stata’s graph schemes, palettes providing colors found in user-contributed schemes, and collections containing sets of palettes found in the literature.

3.1 Stata palettes

The Stata palettes are named after the schemes in which the colors are used. The palettes are:

- s1: 15 colors as in Stata’s s1color scheme
- sir: 15 colors as in Stata’s sircolor scheme
- s2: 15 colors as in Stata’s s2color scheme (the default palette)
- economist: 15 colors as in Stata’s economist scheme
- mono: 15 gray scales as in Stata’s monochrome schemes

Palette s2 is the default used by colorpalette if no palette is specified. The left panel in Figure 1 displays an overview of the palettes.

3.2 User-contributed palettes

Stata users have contributed various scheme files in which alternative sets of colors are used, typically available from the Stata Journal site or from the SSC Archive. The following palettes have been constructed after some of these contributions.

- cblind: 9 colorblind-friendly colors suggested by Okabe and Ito (2002), including an additional gray as suggested at www.cookbook-r.com. The same colors are also used (in different order and using gs10 for gray) in the plotplainblind and plottigblind schemes by Bischof (2017b).
- plottig: 15 colors used for plots 1 to 15 in the plottig scheme by Bischof (2017b). Most of these colors are the same as the colors produced by the hue color generator with default options (see below), although in different order.
- 538: 6 colors used for plots 1 to 6 and 7 colors used for background, labels, axes, and confidence areas in the 538 scheme by Bischof (2017a). The palette replicates colors used at fivethirtyeight.com.
- mrc: 7 colors used for plots 1 to 7 in the mrc scheme by Morris (2013). These are colors according to guidelines by the UK Medical Research Council.
Color palettes for Stata graphics

Figure 1: Stata palettes and user-contributed palettes

<table>
<thead>
<tr>
<th>Stata palettes</th>
<th>User palettes</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>cblind</td>
</tr>
<tr>
<td>s1r</td>
<td>plottig</td>
</tr>
<tr>
<td>s2</td>
<td>538</td>
</tr>
<tr>
<td>economist</td>
<td>mrc</td>
</tr>
<tr>
<td>mono</td>
<td>tfl</td>
</tr>
<tr>
<td></td>
<td>burd</td>
</tr>
<tr>
<td></td>
<td>lean</td>
</tr>
</tbody>
</table>

The right panel in Figure 1 displays an overview of these palettes.

3.3 Collections

ColorBrewer

ColorBrewer is a set of color schemes developed by Brewer et al. (2003; also see Brewer 2016). For more information on ColorBrewer also see colorbrewer2.org. The syntax for the ColorBrewer palettes is

\[ \text{scheme} [, \text{cmyk} \ \text{palette\_options} ] \]

where \text{palette\_options} are general palette options as described above, \text{cmyk} requests the CMYK variant of the colors instead of the RGB variant, and \text{scheme} is one of the following:

1. The colors are licensed under Apache License Version 2.0; see the copyright notes at [www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer_updates.html](http://www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer_updates.html). The RGB values for the implementation of the colors in \text{colorpalette} have been taken from the Excel spreadsheet provided at [www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer_RGB.html](http://www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer_RGB.html). The CMYK values have been taken from file \text{cb.csv} provided at [github.com/axismaps/colorbrewer](http://github.com/axismaps/colorbrewer). ColorBrewer palettes for Stata have also been provided by Gomez (2015) and by Buchanan (2015).
Qualitative schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Description</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accent</td>
<td>8 accented colors for qualitative data</td>
<td>Dark2 8 dark colors for qualitative data</td>
</tr>
<tr>
<td>Paired</td>
<td>12 paired colors for qualitative data</td>
<td>Pastel1 9 pastel colors for qualitative data</td>
</tr>
<tr>
<td>Pastel2</td>
<td>8 pastel colors for qualitative data</td>
<td>Set1 9 colors for qualitative data</td>
</tr>
<tr>
<td>Set2</td>
<td>8 colors for qualitative data</td>
<td>Set3 12 colors for qualitative data</td>
</tr>
</tbody>
</table>

Single-hue sequential schemes (3–9 colors)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blues</td>
<td>light blue to blue</td>
</tr>
<tr>
<td>Greys</td>
<td>light gray to gray</td>
</tr>
<tr>
<td>Purples</td>
<td>light purple to purple</td>
</tr>
</tbody>
</table>

Multi-hue sequential schemes (3–9 colors)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuGn</td>
<td>light blue to green</td>
</tr>
<tr>
<td>GnBu</td>
<td>light green to blue</td>
</tr>
<tr>
<td>PuBu</td>
<td>light purple to blue</td>
</tr>
<tr>
<td>PuRd</td>
<td>light purple to red</td>
</tr>
<tr>
<td>Y1Gn</td>
<td>light yellow to green</td>
</tr>
<tr>
<td>Y1OrBr</td>
<td>light yellow over orange to brown</td>
</tr>
</tbody>
</table>

Diverging schemes (3–11 colors)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrBG</td>
<td>brown to green, light gray mid</td>
</tr>
<tr>
<td>PRGn</td>
<td>purple to green, light gray mid</td>
</tr>
<tr>
<td>RdBu</td>
<td>red to blue, light gray mid</td>
</tr>
<tr>
<td>RdY1Bu</td>
<td>red to blue, yellow mid</td>
</tr>
<tr>
<td>Spectral</td>
<td>red – orange – yellow – green – blue</td>
</tr>
</tbody>
</table>

Figure 2 display the schemes (using the maximum number of colors for those schemes that come in different sizes).

**Semantic colors by Lin et al.**

The *lin* collection provides semantic color schemes suggested by Lin et al. (2013). The syntax is

```
lin [, scheme _algorithm_ palette_options ]
```

where *palette_options* are general palette options as discussed above and *scheme* is one of the following:

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableau</td>
<td>20 categorical colors; the default</td>
</tr>
<tr>
<td>carcolor</td>
<td>6 car colors</td>
</tr>
<tr>
<td>features</td>
<td>5 feature colors</td>
</tr>
<tr>
<td>fruits</td>
<td>7 fruit colors</td>
</tr>
<tr>
<td>drinks</td>
<td>7 drinks colors</td>
</tr>
<tr>
<td>food</td>
<td>7 food colors</td>
</tr>
<tr>
<td>activities</td>
<td>5 activity colors</td>
</tr>
<tr>
<td>vegetables</td>
<td>7 vegetable colors</td>
</tr>
<tr>
<td>brands</td>
<td>7 brands colors</td>
</tr>
</tbody>
</table>

Option *algorithm* requests algorithm selected colors. The default is to return the colors selected by Turkers (in case of *carcolor, food, features, activities*) or by the expert (in case of *fruits, vegetables, drinks, brands*). Option *algorithm* has no effect for *tableau*. Figure 3 display the schemes.

---

2. The values of the semantic colors have been taken from the source code of the brewscheme package by Buchanan (2015) (*brewextra.ado*, version 1.0.0, 21 March 2016); the values of the *tableau* palette have been taken from code provided by Lin et al. at github.com/StanfordHCl/semantics-colors.
Color palettes for Stata graphics

Figure 2: ColorBrewer schemes

Figure 3: Semantic color schemes by Lin et al. (2013)
Color schemes by Paul Tol

The ptol collection provides color schemes as suggested by Tol (2012). The syntax is

```plaintext
ptol [, scheme palette_options ]
```

where `palette_options` are general palette options as discussed above and `scheme` is one of the following (displayed for for different numbers of colors in the upper left panel in Figure 4).

- **qualitative**: 1–12 qualitative colors; the default
- **diverging**: 3–11 diverging colors; very similar to reverse RdYlBu from ColorBrewer
- **rainbow**: 4–12 rainbow colors

D3.js

The d3 collection provides color schemes from d3js.org, using the color values found at github.com/d3/d3-scale/blob/master/README.md#category-scales. The syntax is

```plaintext
d3 [, scheme palette_options ]
```

where `palette_options` are general palette options as discussed above and `scheme` is one of the following (displayed in the upper right panel in Figure 4).

- **10**: 10 categorical colors; the default; same as the first 10 colors in the tableau scheme of the lin collection
- **20**: 20 categorical colors in pairs; same colors as in the tableau scheme of the lin collection, but in different order
- **20b**: 20 categorical colors in groups of four
- **20c**: 20 categorical colors in groups of four

Colors schemes from spmap

The spmap collection provides color schemes from the spmap package by Pisati (2007). The implementation is based on code from spmap.color.ado (version 1.3.0, 13 March 2017). The syntax is:

```plaintext
spmap [, scheme palette_options ]
```

where `palette_options` are general palette options as discussed above and `scheme` is one of the following (displayed for n(16) in the lower left panel in Figure 4).

- **blues**: light blue to blue (2–99 colors); the default
- **greys**: light gray to black (2–99 colors)
- **rainbow**: 2–99 rainbow colors
- **terrain**: 2–16 terrain colors
- **greens**: light green to green (2–99 colors)
- **reds**: light red to red (2–99 colors)
- **heat**: 2–16 heat colors
- **topological**: 2–16 topological colors
Swiss Federal Statistical Office colors

The `sfso` collection provides color schemes by the Swiss Federal Statistical Office (using hex and CMYK codes found in Bundesamt für Statistik 2017). The syntax is

```
sfso , scheme cmyk palette.options
```

where `palette.options` are general palette options as discussed above and `scheme` is one of the following (displayed in the lower right panel in Figure 4).

Sequential schemes

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>brown</td>
<td>dark brown to light brown</td>
</tr>
<tr>
<td>red</td>
<td>dark red to light red</td>
</tr>
<tr>
<td>purple</td>
<td>dark purple to light purple</td>
</tr>
<tr>
<td>blue</td>
<td>dark blue to light blue; the default</td>
</tr>
<tr>
<td>turquoise</td>
<td>dark turquoise to light turquoise</td>
</tr>
<tr>
<td>olive</td>
<td>dark olive to light olive</td>
</tr>
<tr>
<td>orange</td>
<td>dark orange to light orange</td>
</tr>
<tr>
<td>pink</td>
<td>dark pink to light pink</td>
</tr>
<tr>
<td>violet</td>
<td>dark violet to light violet</td>
</tr>
<tr>
<td>ltblue</td>
<td>lighter version of blue</td>
</tr>
<tr>
<td>green</td>
<td>dark green to light green</td>
</tr>
<tr>
<td>black</td>
<td>dark gray to light gray</td>
</tr>
</tbody>
</table>

Semantic schemes

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parties</td>
<td>colors used for Swiss parties</td>
</tr>
<tr>
<td>languages</td>
<td>colors used for languages</td>
</tr>
<tr>
<td>votes</td>
<td>colors used for results from votes</td>
</tr>
</tbody>
</table>

Option `cmyk` requests the CMYK variant of a scheme. The default is to use the RGB variant.
4 Color generators

4.1 Evenly spaced HCL hues

The `hue` palette implements an algorithm that generates HCL colors with evenly spaced hues. The palette has been modeled after function `hue_pal()` from R’s `scales` package by Hadley Wickham (see github.com/hadley/scales). This is the default color scheme used by R’s `ggplot2` for categorical data (see ggplot2.tidyverse.org/reference/scale_hue.html). The `hue` palette with default options produces the same colors as the `intense` scheme of the `hcl` color generator (see below). The syntax of the `hue` color generator is

```
hue [, hue_options palette_options ]
```

where `palette_options` are general palette options as discussed above and `hue_options` are:

- `hue(h1, h2)` sets the range of hues on the 360 degree color wheel. The default is `hue(15, 375)`. If the difference between start and end is a multiple of 360, end will be reduced by `360/n`, where `n` is the number of requested colors (so that the space between the last and the first color is the same as between the other colors).

- `chroma(c)` sets the colorfulness (color intensity), with `c ≥ 0`. The default is `chroma(100)`.

- `luminance(l)` sets the brightness (amount of gray), with `l ∈ [0, 100]`. The default is `luminance(65)`.

- `direction(#)` determines the direction to travel around the color wheel. `direction(1)`, the default, travels clockwise; `direction(-1)` travels counter-clockwise.

The following graph illustrates how the colors change depending on option `n()`:

![Color palette graph](image-url)
4.2 HCL color generator

The hcl palette is a HCL color generator (Hue-Chroma-Luminance). The implementation is based on R’s colorspace package by Ihaka et al. (2016); also see Zeileis et al. (2009) and hclwizard.org.

Let \( h_1 \) and \( h_2 \) be two hues on the 360 degree color wheel, \( c_1 \) and \( c_2 \) two chroma levels, \( l_1 \) and \( l_2 \) two luminance levels, \( p_1 \) and \( p_2 \) two power parameters, and \( i \) an index from 1 to \( n \), where \( n \) is the number of requested colors. The HCL colors are then generated according to the following formulas.

Qualitative schemes:

\[
H = h_1 + j(h_2 - h_1), \quad C = c_1, \quad L = l_1, \quad j = \frac{i - 1}{n - 1}
\]

Sequential schemes:

\[
H = h_2 - j(h_2 - h_1), \quad C = c_2 - j^{p_1}(c_2 - c_1), \quad L = l_2 - j^{p_2}(l_2 - l_1), \quad j = \frac{n - i}{n - 1}
\]

Diverging schemes:

\[
H = \begin{cases} 
    h_1 & \text{if } j > 0 \\
    h_2 & \text{else}
\end{cases}, \quad C = |j|^{p_1}c_1, \quad L = l_2 - |j|^{p_2}(l_2 - l_1), \quad j = \frac{n - 2j + 1}{n - 1}
\]

The syntax of the hcl color generator is

```
hcl [, scheme hcl_options palette_options ]
```

where palette_options are general palette options as discussed above and scheme selects the type of scheme and the default parameters according to the following overview:

<table>
<thead>
<tr>
<th>Qualitative</th>
<th>( h_1 )</th>
<th>( h_2 )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( l_1 )</th>
<th>( l_2 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>Qualitative</th>
<th>( h_1 )</th>
<th>( h_2 )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( l_1 )</th>
<th>( l_2 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>qualitative</td>
<td>15</td>
<td>( h^* )</td>
<td>60</td>
<td>70</td>
<td>( \ldots )</td>
<td>intense</td>
<td>15</td>
<td>( h^* )</td>
<td>100</td>
<td>65</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dark</td>
<td>15</td>
<td>( h^* )</td>
<td>80</td>
<td>60</td>
<td>( \ldots )</td>
<td>light</td>
<td>15</td>
<td>( h^* )</td>
<td>50</td>
<td>80</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pastel</td>
<td>15</td>
<td>( h^* )</td>
<td>35</td>
<td>85</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequential</td>
<td>260</td>
<td>( h_1 )</td>
<td>80</td>
<td>10</td>
<td>25</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>260</td>
<td>( h_1 )</td>
<td>80</td>
<td>10</td>
<td>25</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>greens</td>
<td>145</td>
<td>125</td>
<td>80</td>
<td>10</td>
<td>25</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>0</td>
<td>15</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>oranges</td>
<td>40</td>
<td>( h_1 )</td>
<td>100</td>
<td>10</td>
<td>50</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>280</td>
<td>( h_1 )</td>
<td>70</td>
<td>10</td>
<td>20</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>reds</td>
<td>10</td>
<td>20</td>
<td>80</td>
<td>10</td>
<td>25</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>0</td>
<td>100</td>
<td>30</td>
<td>50</td>
<td>90</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>heat2</td>
<td>0</td>
<td>90</td>
<td>80</td>
<td>30</td>
<td>30</td>
<td>90</td>
<td>.2</td>
<td>2</td>
<td>130</td>
<td>0</td>
<td>80</td>
<td>0</td>
<td>60</td>
<td>95</td>
<td>.1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>terrain2</td>
<td>130</td>
<td>30</td>
<td>65</td>
<td>0</td>
<td>45</td>
<td>90</td>
<td>.5</td>
<td>1.5</td>
<td>( \text{viridis} )</td>
<td>300</td>
<td>75</td>
<td>35</td>
<td>95</td>
<td>15</td>
<td>90</td>
<td>.8</td>
<td>1.2</td>
</tr>
<tr>
<td>plasma</td>
<td>100</td>
<td>( h_1 )</td>
<td>60</td>
<td>100</td>
<td>15</td>
<td>95</td>
<td>2</td>
<td>.9</td>
<td>( \text{redblue} )</td>
<td>0</td>
<td>-100</td>
<td>80</td>
<td>40</td>
<td>40</td>
<td>75</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Diverging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diverging</td>
<td>260</td>
<td>0</td>
<td>80</td>
<td>30</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>260</td>
<td>0</td>
<td>80</td>
<td>30</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bluered2</td>
<td>260</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>180</td>
<td>330</td>
<td>60</td>
<td>75</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greenorange</td>
<td>130</td>
<td>45</td>
<td>100</td>
<td>70</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>55</td>
<td>160</td>
<td>60</td>
<td>35</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pinkgreen</td>
<td>340</td>
<td>128</td>
<td>90</td>
<td>35</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td>300</td>
<td>128</td>
<td>60</td>
<td>30</td>
<td>95</td>
<td>1</td>
<td>( p_1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
hcl\_options are:

\texttt{hue(h1 [h2])} overwrites the default values for \(h_1\) and \(h_2\) that determine the range of hues on the 360 degree color wheel.

\texttt{chroma(c1 [c2])} overwrites the default values for \(c_1\) and \(c_2\), \(c_i \geq 0\), that determine the colorfulness (color intensity).

\texttt{luminance(l1 [l2])} overwrites the default values for \(l_1\) and \(l_2\), \(l_i \in [0, 100]\), that determine the brightness (amount of gray).

\texttt{power(p1 [p2])} overwrites the default values for \(p_1\) and \(p_2\), \(p_i > 0\), that determine the shape of the transition between chroma and luminance levels. For linear transitions, set \(p_i = 1\); \(p_i > 1\) makes the transition faster, \(p_i < 1\) makes the transition slower.

The left panel of Figure 5 displays the predefined HCL schemes with default parameters for \(n = 15\).

### 4.3 HSV color generator

The \texttt{hsv} palette is a HSV color generator (Hue-Saturation-Value). The implementation is partially based on R’s \texttt{grDevices} package (which is part of the R core) and partially on \texttt{colorspace} by Ihaka et al. (2016).

Let \(h_1\) and \(h_2\) be two hues on the 360 degree color wheel, \(s_1\) and \(s_2\) two saturation levels, \(v_1\) and \(v_2\) two value levels, \(p_1\) and \(p_2\) two power parameters, and \(i\) an index from 1 to \(n\), where \(n\) is the number of requested colors. The HSV colors are then generated according to the following formulas.

Qualitative schemes:

\[
H = h_1 + j(h_2 - h_1), \quad S = s_1, \quad V = v_1, \quad j = \frac{i - 1}{n - 1}
\]

Sequential schemes:

\[
H = h_2 - j(h_2 - h_1), \quad S = s_2 - j^{p_1}(s_2 - s_1), \quad V = v_2 - j^{p_2}(v_2 - v_1), \quad j = \frac{n - i}{n - 1}
\]

Diverging schemes:

\[
H = \begin{cases} 
  h_1 & \text{if } j > 0 \\
  h_2 & \text{else}
\end{cases}, \quad S = |j|^{p_1}s_1, \quad V = v_2 - |j|^{p_2}(v_2 - v_1), \quad j = \frac{n - 2j + 1}{n - 1}
\]

The syntax of the \texttt{hsv} color generator is

\texttt{hsv \[ \text{, \ scheme \ hsv\_options \ palette\_options } \]}

where \texttt{palette\_options} are general palette options as discussed above and \texttt{scheme} selects the type of scheme and the default parameters according to the following overview:
### Color palettes for Stata graphics

<table>
<thead>
<tr>
<th>Qualitative</th>
<th>( h_1 )</th>
<th>( h_2 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>qualitative</td>
<td>15</td>
<td>( h^* )</td>
<td>.4</td>
<td>-.85</td>
<td>-.85</td>
<td>-.85</td>
<td>-.85</td>
<td>-.85</td>
</tr>
<tr>
<td>dark</td>
<td>15</td>
<td>( h^* )</td>
<td>.6</td>
<td>-.7</td>
<td>-.7</td>
<td>-.7</td>
<td>-.7</td>
<td>-.7</td>
</tr>
<tr>
<td>pastel</td>
<td>15</td>
<td>( h^* )</td>
<td>.2</td>
<td>-.9</td>
<td>-.9</td>
<td>-.9</td>
<td>-.9</td>
<td>-.9</td>
</tr>
</tbody>
</table>

### Sequential

<table>
<thead>
<tr>
<th>sequential</th>
<th>( h_1 )</th>
<th>( h_2 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>blues</td>
<td>240</td>
<td>.8</td>
<td>.05</td>
<td>.6</td>
<td>1</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>greens</td>
<td>140</td>
<td>120</td>
<td>1</td>
<td>.1</td>
<td>.3</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>oranges</td>
<td>30</td>
<td>( h_1 )</td>
<td>1</td>
<td>.1</td>
<td>.9</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>reds</td>
<td>0</td>
<td>20</td>
<td>1</td>
<td>.1</td>
<td>.6</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>terrain</td>
<td>120</td>
<td>0</td>
<td>1</td>
<td>.05</td>
<td>.95</td>
<td>0.7</td>
<td>1.5</td>
<td></td>
</tr>
</tbody>
</table>

### Diverging

<table>
<thead>
<tr>
<th>diverging</th>
<th>( h_1 )</th>
<th>( h_2 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>bluered</td>
<td>240</td>
<td>0</td>
<td>-.8</td>
<td>.6</td>
<td>.95</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>bluered2</td>
<td>240</td>
<td>0</td>
<td>-.6</td>
<td>.8</td>
<td>.95</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>greenorange</td>
<td>130</td>
<td>40</td>
<td>-.9</td>
<td>.8</td>
<td>.95</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
<tr>
<td>pinkgreen</td>
<td>330</td>
<td>120</td>
<td>-.9</td>
<td>.6</td>
<td>.95</td>
<td>1.2</td>
<td>( p_1 )</td>
<td></td>
</tr>
</tbody>
</table>

### hsv_options are:

- \(_{\text{hue}}(h_1 \ [h_2])_\) overwrites the default values for \( h_1 \) and \( h_2 \) that determine the range of hues on the 360 degree color wheel.

- \(_{\text{saturation}}(s_1 \ [s_2])_\) overwrites the default values for \( s_1 \) and \( s_2 \), \( s_i \in [0,1] \), that determine the colorfulness (color intensity).

- \(_{\text{value}}(v_1 \ [v_2])_\) overwrites the default values for \( v_1 \) and \( v_2 \), \( v_i \in [0,1] \), that determine the brightness (amount of gray).

- \(_{\text{power}}(p_1 \ [p_2])_\) overwrites the default values for \( p_1 \) and \( p_2 \), \( p_i > 0 \), that determine the shape of the transition between chroma and luminance levels. For linear transitions, set \( p_i = 1 \); \( p_i > 1 \) makes the transition faster, \( p_i < 1 \) makes the transition slower.

The right panel of Figure 5 displays the predefined HSV schemes with default parameters for \( n = 15 \).

### 5 Appendix: Symbol palettes and line pattern palettes

The \texttt{palettes} package also contains commands for symbol palettes and line pattern palettes. Their syntax and basic functionality is similar to the command for color palettes; see \texttt{help symbolpalette} and \texttt{help linepalette}. Figure 6 display an overview of the available named palettes.

### 6 References


———. 2017b. New graphic schemes for Stata: plotplain and plottig. \textit{The Stata Journal}
Figure 5: HCL and HCV color schemes
Color palettes for Stata graphics

![Symbol palettes and line pattern palettes](image)

Figure 6: Symbol palettes and line pattern palettes

17(3): 748–759.


**About the author**

Ben Jann is Professor of Sociology at the University of Bern, Switzerland. His research interests include social-science methodology, statistics, social stratification, and labor market sociology. He is principle investigator of TREE, a large-scale multi-cohort panel study in Switzerland on transitions from education to employment (www.tree.unibe.ch).