The Stata Journal (2018)
18, Number 4, pp. 765–785

Color palettes for Stata graphics

Ben Jann
University of Bern
Bern, Switzerland
ben.jann@soz.unibe.ch

Abstract. In this article, I introduce the `colorpalette` command, which provides many color palettes and color generators for use in Stata graphics. It supports 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, and HSV and HCL color generators. As a by-product, I also introduce commands for marker-symbol and line-pattern palettes.

Keywords: gr0075, palettes, colorpalette, symbolpalette, linepalette, graph, graphics, color, color spaces

This article has extensive use of colors, so the electronic copy has been published in color while the printed copy is in monochrome. If you are reading the printed copy and are having trouble following the text, you may want to switch to the electronic copy.

1 Introduction

Stata features a set of about 50 named colors that can be used in graphs (see [G-4] ***colorstyle***). Given the diverse needs of users, a set of 50 predefined colors is rather limited. Alternative colors are supported but must be specified by their RGB, CMYK, or HSV values. To increase the number of easily accessible colors, the new `colorpalette` command provides many 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 of intensity or opacity levels.

The primary purpose of `colorpalette` is to provide color palettes, color generators, and additional color input formats for use with `grstyle set`, which customizes Stata graphics (Jann 2018). `grstyle set` runs `colorpalette` automatically so that users typically do not have to call `colorpalette` directly. However, manually calling `colorpalette` can be useful to generate a quick overview picture of one or several palettes (see sections 2.1 and 2.3). Furthermore, `colorpalette` can also be used independently of `grstyle set` to retrieve colors and then pass them through to a subsequent graph command (for an example, see section 2.2).

By-products of `colorpalette` are two additional commands, `symbolpalette` and `linepalette`, that provide palettes of marker symbols and line pattern. I briefly present these commands in the appendix.

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 displayed by default in a graph. The syntax is

`colorpalette` [*argument*] [`,` *palette_options graph_options*]

where *argument* is

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-4] ***colorstyle***; HCL values are 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 are 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

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

where *pspec* is

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 (`hue`, `hcl`, `hsv`, `ptol`, sequential and diverging ColorBrewer palettes) 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. The values in *numlist* must be between 0 and 255. Values below 1 make the colors lighter; values larger than 1

make the colors darker. Specify multiple values to use different adjustments across the selected palette elements. The list of adjustments will be recycled if it is shorter than the list of selected palette elements. Likewise, palette elements will be recycled if the list of adjustments is longer than the palette.

`opacity(`*numlist*`)` sets the opacity level or levels (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-3] ***twoway_options***.

Additional graph options for syntax 1

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

`nograph` suppresses the graph.

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.

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

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

Stored results

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

Scalars:

`r(n)`	number of colors

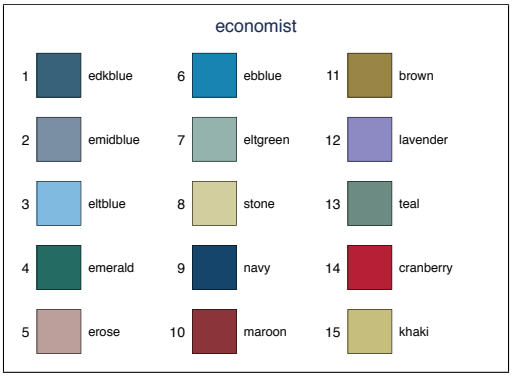
Macros:

`r(ptype)`	`color`	`r(p)`	space-separated list of colors
`r(pname)`	name of palette or `custom`	`r(p#)`	#th color
`r(pnote)`	palette note (if available)	`r(p#info)`	info on #th color (if available)

2.1 Viewing a palette (syntax 1)

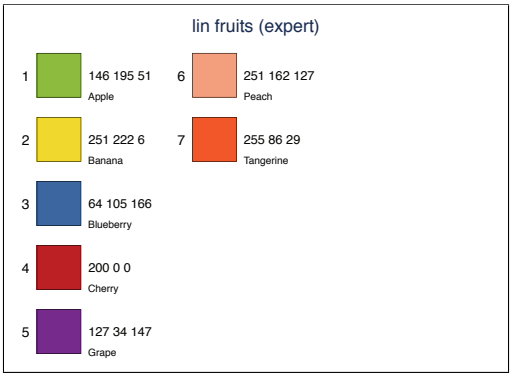
To display a single palette, type `colorpalette` followed by the name of the palette. For example, to view the `economist` palette, type

```
. colorpalette economist
```



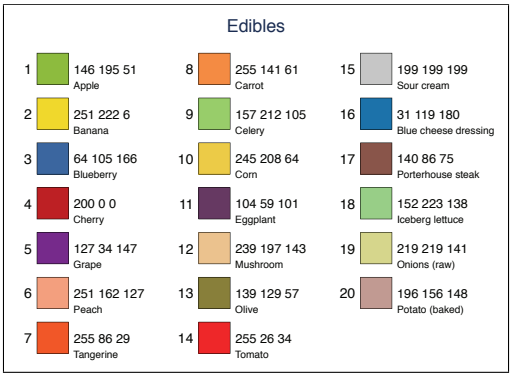
The graph produced by `colorpalette` displays the colors, their names or color codes, and possibly some additional information. Here is an example of a semantic palette by Lin et al. (2013) with RGB codes and labels:

```
. colorpalette lin, fruits
```



You can also 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):

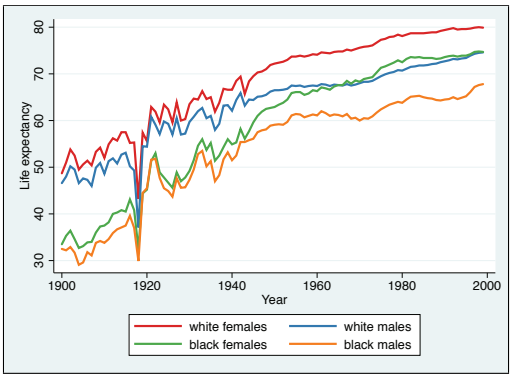
```
. colorpalette lin, fruits
>            / lin, vegetable
>            / lin, food select(1/6)
>              title("Edibles")
```



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 (the option `nograph` is specified to prevent `colorpalette` from displaying the palette):

```
. sysuse uslifeexp, clear
(U.S. life expectancy, 1900-1999)
. label variable le_wfemale "white females"
. label variable le_wmale   "white males"
. label variable le_bfemale "black females"
. label variable le_bmale   "black males"
. colorpalette Set1, select(1/3 5)
>     nograph
. line le_wfemale le_wmale
>      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)´"´
```

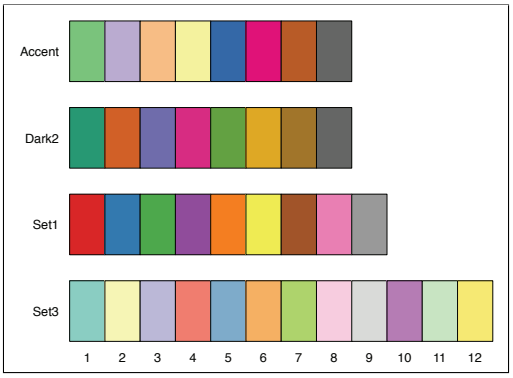
would copy the list of colors to local macro `mycolor`. You could then use the colors in subsequent graph commands by typing `` `mycolors´ ``.

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)

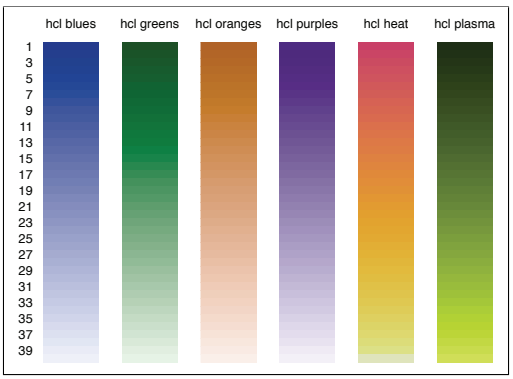
To display an overview of multiple palettes in a single graph, type `colorpalette`, a colon, and a list of palettes separated by forward slashes. The following example displays some of the categorical palettes from ColorBrewer (Brewer 2015; Brewer, Hatchard, and Harrower 2003):

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



The 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:

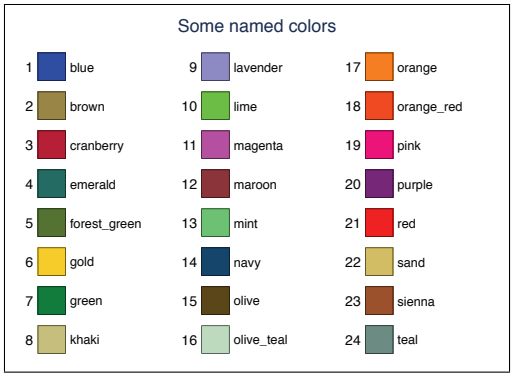
```
. colorpalette, vertical n(40):
>     hcl, blues / hcl, greens /
>     hcl, oranges / hcl, purples /
>     hcl, heat / hcl, plasma
```



2.4 Specifying a custom list of colors

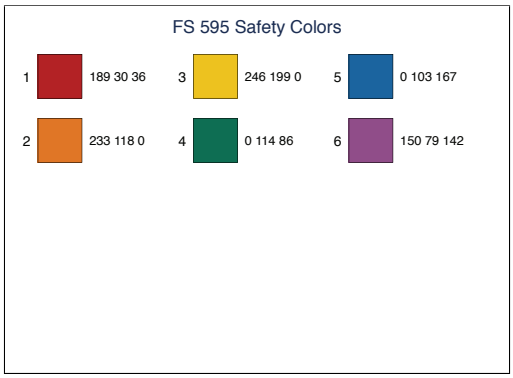
Instead of using a named palette, you can provide a custom palette by specifying a list of *colorstyles* (named colors, RGB values, CMYK values, or HSV values; see [G-4] ***colorstyle***). Here is an example displaying some of Stata's named colors:

```
. 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-4] ***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 https://www.w3schools.com/colors/colors_fs595.asp:

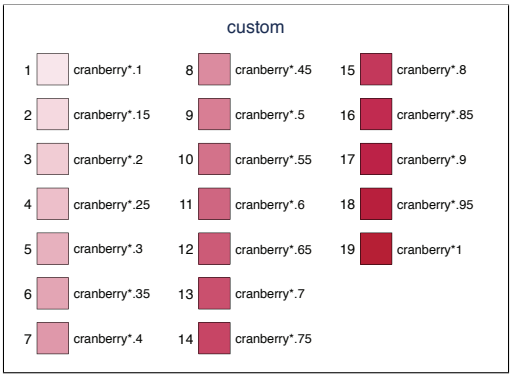
```
. colorpalette #bd1e24 #e97600
>     #f6c700 #007256 #0067a7
>     #964f8e, rows(2)
>     title(FS 595 Safety Colors)
```



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 arguments (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 you to create colors over a range of intensities or opacity levels; see the following example:

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

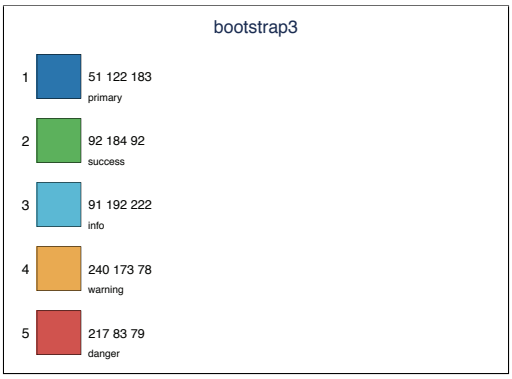


2.6 Custom palettes

If you want to create a personal, 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 (for example, RGB values or hex codes) as a comma-separated list in local macro `P`. You may also provide a comma-separated list of descriptions in local macro `I`.

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

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



More complicated definitions of palettes that account for 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 community-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
`s1r`	15 colors as in Stata's `s1rcolor` 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.

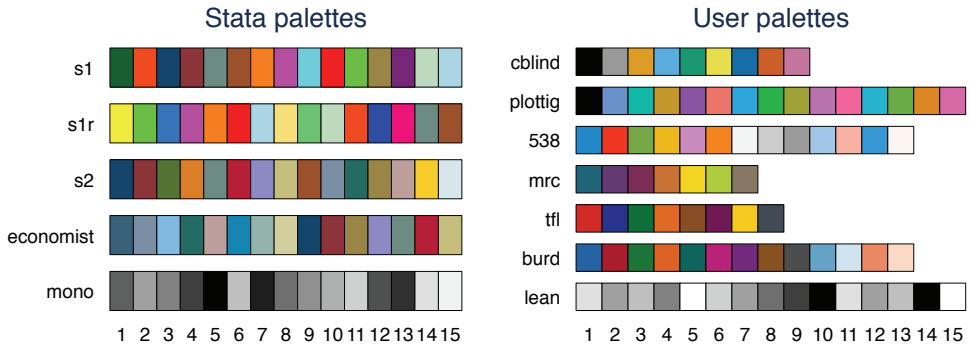


Figure 1. Stata palettes and community-contributed palettes

3.2 Community-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 Statistical Software Components archive. The following palettes have been constructed after some of these contributions.

`cblind`	Nine colorblind-friendly colors suggested by Okabe and Ito (2002), including an additional gray as suggested at http://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`	Fifteen 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`	Six colors used for plots 1 to 6 and seven colors used for background, labels, axes, and confidence areas in the `538` scheme by Bischof (2017a). The palette replicates colors used at https://fivethirtyeight.com.
`mrc`	Seven 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.
`tfl`	Eight colors used for plots 1 to 8 in the `tfl` scheme by Morris (2015). The palette replicates Transport for London's corporate colors.
`burd`	Nine colors used for plots 1 to 9 and four colors used for confidence areas in the `burd` scheme by Briatte (2013). The first nine colors are a selection of colors from various ColorBrewer schemes.
`lean`	Fifteen gray scales used for areas in plots 1 to 15 in schemes `lean1` and `lean2` by Juul (2003).

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, Hatchard, and Harrower ([2003]; also see Brewer [2015]). For more information on ColorBrewer, also see http://colorbrewer2.org.[1] The syntax for the ColorBrewer palettes is

scheme [, `cmyk` *palette_options*]

where *palette_options* are general palette options as described above, `cmyk` requests the CMYK variant of the colors instead of the RGB variant, and `scheme` is one of the following:

Qualitative schemes

`Accent`	8 accented colors for qualitative data	`Pastel2`	8 pastel colors for qualitative data
`Dark2`	8 dark colors for qualitative data	`Set1`	9 colors for qualitative data
`Paired`	12 paired colors for qualitative data	`Set2`	8 colors for qualitative data
`Pastel1`	9 pastel colors for qualitative data	`Set3`	12 colors for qualitative data

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

Single-hue sequential schemes (3–9 colors)

`Blues`	light blue to blue	`Oranges`	light orange to orange
`Greens`	light green to green	`Purples`	light purple to purple
`Greys`	light gray to gray	`Reds`	light red to red

Multihue sequential schemes (3–9 colors)

`BuGn`	light blue to green	`PuRd`	light purple to red
`BuPu`	light blue to purple	`RdPu`	light red to purple
`GnBu`	light green to blue	`YlGn`	light yellow to green
`OrRd`	light orange to red	`YlGnBu`	light yellow over green to blue
`PuBu`	light purple to blue	`YlOrBr`	light yellow over orange to brown
`PuBuGn`	light purple over blue to green	`YlOrRd`	light yellow over orange to red

Diverging schemes (3–11 colors)

`BrBG`	brown to green, light gray mid	`RdGy`	red to gray, white mid
`PiYG`	pink to green, light gray mid	`RdYlBu`	red to blue, yellow mid
`PRGn`	purple to green, light gray mid	`RdYlGn`	red to green, yellow mid
`PuOr`	purple to orange, light gray mid	`Spectral`	red – orange – yellow – green – blue
`RdBu`	red to blue, light gray mid		

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

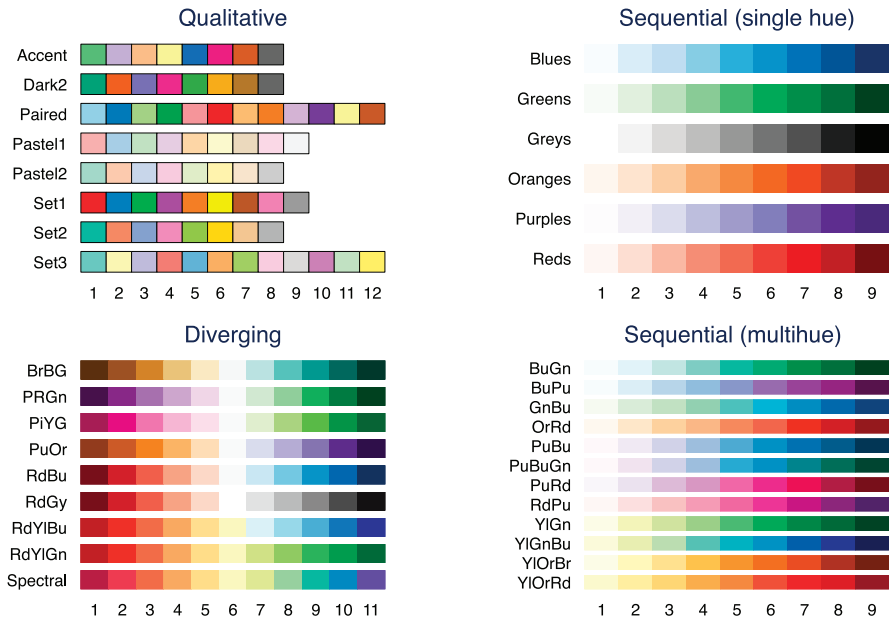


Figure 2. ColorBrewer schemes

Semantic colors by Lin et al. (2013)

The `lin` collection provides semantic color schemes suggested by Lin et al. (2013).[2] 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:

`tableau`	20 categorical colors; the default	`fruits`	7 fruit colors
`carcolor`	6 car colors	`vegetables`	7 vegetable colors
`food`	7 food colors	`drinks`	7 drinks colors
`features`	5 feature colors	`brands`	7 brands colors
`activities`	5 activity colors		

The 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`). The option `algorithm` has no effect for `tableau`. Figure 3 displays the schemes.

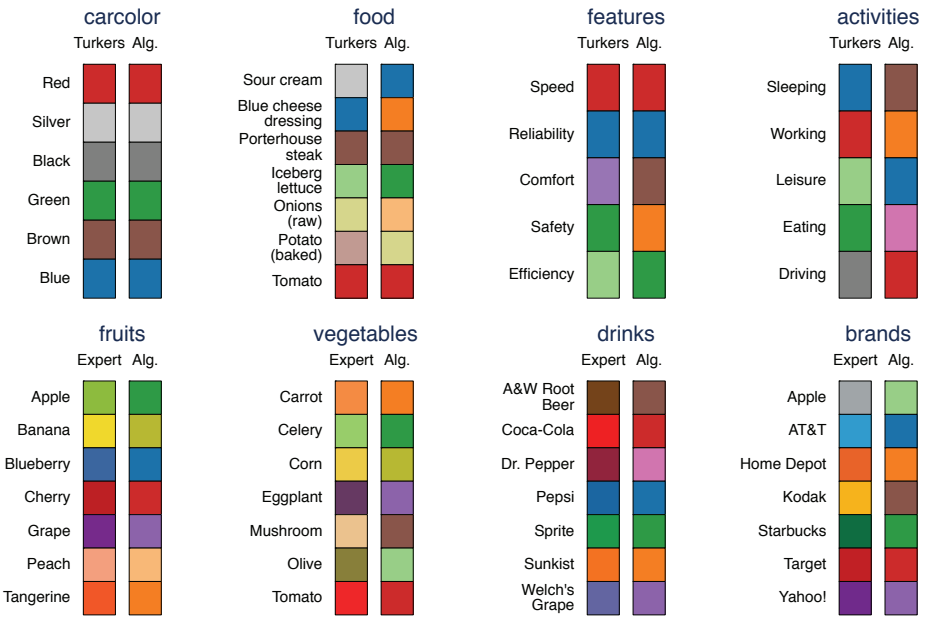


Figure 3. Semantic color schemes by Lin et al. (2013)

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 https://github.com/StanfordHCI/semantic-colors.

Color schemes by Tol (2018)

The `ptol` collection provides color schemes as suggested by Tol (2018). The syntax is

`ptol` [, *scheme palette_options*]

where *palette_options* are general palette options as discussed above and *scheme* is one of the following (displayed 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; similar to reverse `RdYlBu` from ColorBrewer
`rainbow`	4–12 rainbow colors

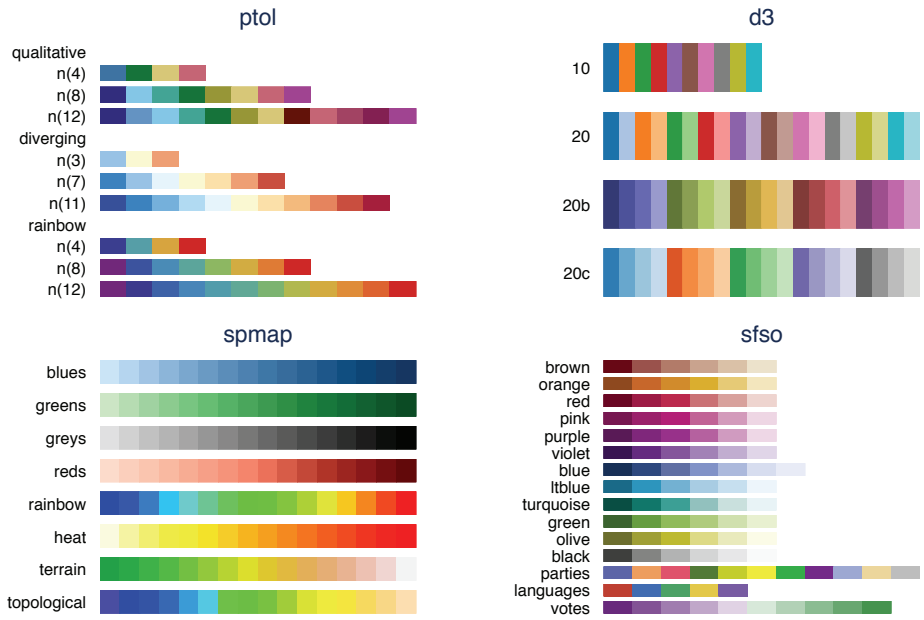


Figure 4. Various palettes

D3.js

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

`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 4
`20c`	20 categorical colors in groups of 4

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

`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	`rainbow`	2–99 rainbow colors
`greens`	light green to green (2–99 colors)	`heat`	2–16 heat colors
`greys`	light gray to black (2–99 colors)	`terrain`	2–16 terrain colors
`reds`	light red to red (2–99 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

`brown`	dark brown to light brown	`blue`	dark blue to light blue; the default
`orange`	dark orange to light orange	`ltblue`	lighter version of `blue`
`red`	dark red to light red	`turquoise`	dark turquoise to light turquoise
`pink`	dark pink to light pink	`green`	dark green to light green
`purple`	dark purple to light purple	`olive`	dark olive to light olive
`violet`	dark violet to light violet	`black`	dark gray to light gray

Semantic schemes

`parties`	colors used for Swiss parties	`votes`	colors used for results from votes
`languages`	colors used for languages		

The 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 http://github.com/hadley/scales). This is the default color scheme used by R's `ggplot2` for categorical data (see http://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 the following:

`hue(`h_1 h_2`)` 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).

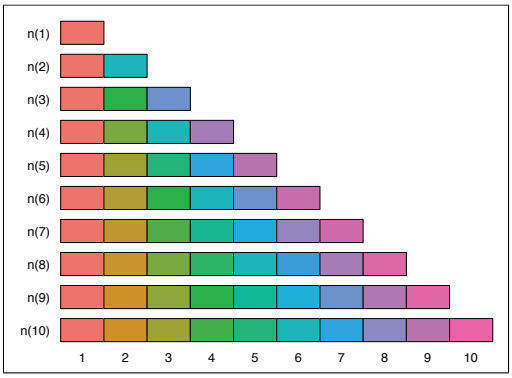
`chroma(`c`)` sets the colorfulness (color intensity), with $c \geq 0$. The default is `chroma(100)`.

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

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

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

```
. colorpalette, plabels(n(1) n(2)
>         n(3) n(4) n(5) n(6) n(7)
>         n(8) n(9) n(10))
>     lcolor(black):
>     hue, n(1) / hue, n(2) /
>     hue, n(3) / hue, n(4) /
>     hue, n(5) / hue, n(6) /
>     hue, n(7) / hue, n(8) /
>     hue, n(9) / hue, n(10)
```



4.2 HCL color generator

The `hcl` palette is an HCL color generator based on R's `colorspace` package by Ihaka et al. (2016); also see Zeileis, Hornik, and Murrell (2009) and http://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:

	h_1	h_2	c_1	c_2	l_1	l_2	p_1	p_2		h_1	h_2	c_1	c_2	l_1	l_2	p_1	p_2
Qualitative																	
`qualitative`	15	h^*	60	–	70	–	–	–	`light`	15	h^*	50	–	80	–	–	–
`intense`	15	h^*	100	–	65	–	–	–	`pastel`	15	h^*	35	–	85	–	–	–
`dark`	15	h^*	80	–	60	–	–	–		with $h^* = h_1 + 360(n-1)/n$							
Sequential																	
`sequential`	260	h_1	80	10	25	95	1	p_1	`heat`	0	90	100	30	50	90	.2	1
`blues`	260	h_1	80	10	25	95	1	p_1	`heat2`	0	90	80	30	30	90	.2	2
`greens`	145	125	80	10	25	95	1	p_1	`terrain`	130	0	80	0	60	95	.1	1
`grays`	0	h_1	0	0	15	95	1	p_1	`terrain2`	130	30	65	0	45	90	.5	1.5
`oranges`	40	h_1	100	10	50	95	1	p_1	`viridis`	300	75	35	95	15	90	.8	1.2
`purples`	280	h_1	70	10	20	95	1	p_1	`plasma`	100	h_1	60	100	15	95	2	.9
`reds`	10	20	80	10	25	95	1	p_1	`redblue`	0	-100	80	40	40	75	1	1
Diverging																	
`diverging`	260	0	80	–	30	95	1	p_1	`greenorange`	130	45	100	–	70	95	1	p_1
`bluered`	260	0	80	–	30	95	1	p_1	`browngreen`	55	160	60	–	35	95	1	p_1
`bluered2`	260	0	100	–	50	95	1	p_1	`pinkgreen`	340	128	90	–	35	95	1	p_1
`bluered3`	180	330	60	–	75	95	1	p_1	`purplegreen`	300	128	60	–	30	95	1	p_1

hcl_options are the following:

`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.

`chroma(`c_1 [c_2]`)` overwrites the default values for c_1 and c_2, with $c_i \geq 0$. c_1 and c_2 determine the colorfulness (color intensity).

`luminance(`l_1 [l_2]`)` overwrites the default values for l_1 and l_2, with $l_i \in [0, 100]$. l_1 and l_2 determine the brightness (amount of gray).

`power(`p_1 [p_2]`)` overwrites the default values for p_1 and p_2, with $p_i > 0$. p_1 and p_2 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 `hsv` palette is an HSV color generator. The implementation is partially based on R's `grDevices` package (which is part of the R core) and partially on `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 `hsv` color generator is

`hsv` [`,` *scheme hsv_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:

	h_1	h_2	s_1	s_2	v_1	v_2	p_1	p_2		h_1	h_2	s_1	s_2	v_1	v_2	p_1	p_2
Qualitative																	
`qualitative`	15	h^*	.4	–	.85	–	–	–	`light`	15	h^*	.3	–	.9	–	–	–
`intense`	15	h^*	.6	–	.9	–	–	–	`pastel`	15	h^*	.2	–	.9	–	–	–
`dark`	15	h^*	.6	–	.7	–	–	–	`rainbow`	15	h^*	1	–	1	–	–	–
										with $h^* = h_1 + 360(n-1)/n$							
Sequential																	
`sequential`	240	h_1	.8	.05	.6	1	1.2	p_1	`purples`	270	h_1	1	.1	.6	1	1.2	p_1
`blues`	240	h_1	.8	.05	.6	1	1.2	p_1	`reds`	0	20	1	.1	.6	1	1.2	p_1
`greens`	140	120	1	.1	.3	1	1.2	p_1	`heat`	0	60	1	.2	1	1	0.3	p_1
`grays`	0	h_1	0	0	.1	.95	1.0	p_1	`terrain`	120	0	1	0	.65	.95	0.7	1.5
`oranges`	30	h_1	1	.1	.9	1	1.2	p_1									
Diverging																	
`diverging`	240	0	–	.8	.6	.95	1.2	p_1	`greenorange`	130	40	–	1	.8	.95	1.2	p_1
`bluered`	240	0	–	.8	.6	.95	1.2	p_1	`browngreen`	40	150	–	.8	.6	.95	1.2	p_1
`bluered2`	240	0	–	.6	.8	.95	1.2	p_1	`pinkgreen`	330	120	–	.9	.6	.95	1.2	p_1
`bluered3`	175	320	–	.6	.8	.95	1.2	p_1	`purplegreen`	290	120	–	.7	.5	.95	1.2	p_1

hsv_options are the following:

`hue(`h_1 $\left[\,h_2\,\right]$`)` overwrites the default values for h_1 and h_2 that determine the range of hues on the 360-degree color wheel.

`saturation(`s_1 $\left[\,s_2\,\right]$`)` overwrites the default values for s_1 and s_2, with $s_i \in [0,1]$. s_1 and s_2 determine the colorfulness (color intensity).

`value(`v_1 $\left[\,v_2\,\right]$`)` overwrites the default values for v_1 and v_2, with $v_i \in [0,1]$. v_1 and v_2 determine the brightness (amount of gray).

`power(`p_1 $\left[\,p_2\,\right]$`)` overwrites the default values for p_1 and p_2, with $p_i > 0$. p_1 and p_2 determine the shape of the transition between saturation and value 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 $n = 15$.

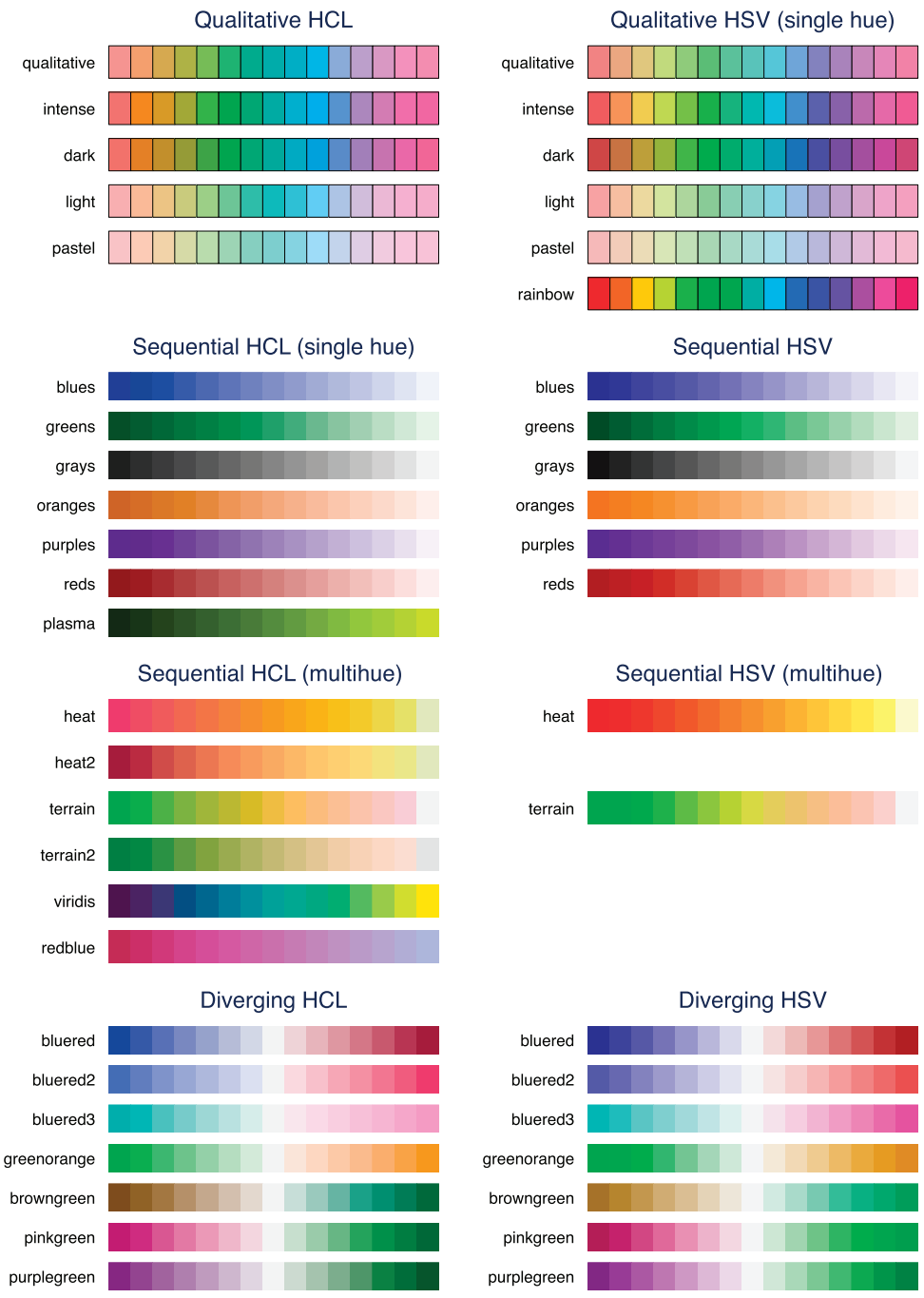


Figure 5. HCL and HSV color schemes

5 References

Bischof, D. 2017a. g538schemes: Stata module to provide graphics schemes for http://fivethirtyeight.com. Statistical Software Components S458404, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458404.html.

———. 2017b. New graphic schemes for Stata: plotplain and plottig. *Stata Journal* 17: 748–759.

Brewer, C. A. 2015. *Designing Better Maps: A Guide for GIS Users*. 2nd ed. Redlands, CA: ESRI Press.

Brewer, C. A., G. W. Hatchard, and M. A. Harrower. 2003. ColorBrewer in print: A catalog of color schemes for maps. *Cartography and Geographic Information Science* 30: 5–32.

Briatte, F. 2013. scheme-burd: Stata module to provide a ColorBrewer-inspired graphics scheme with qualitative and blue-to-red diverging colors. Statistical Software Components S457623, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s457623.html.

Buchanan, W. 2015. brewscheme: Stata module for generating customized graph scheme files. Statistical Software Components S458050, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458050.html.

Bundesamt für Statistik. 2017. Layoutrichtlinien. Gestaltungs und Redaktionsrichtlinien für Publikationen, Tabellen und grafische Assets. Technical Report Version 1.1.1, Bundesamt für Statistik, Neuchâtel.

Gomez, M. 2015. Stata command to generate color schemes. http://github.com/matthieugomez/stata-colorscheme.

Ihaka, R., P. Murrell, K. Hornik, J. C. Fisher, R. Stauffer, and A. Zeileis. 2016. *colorspace: Color space manipulation*. R package version 1.3-2. https://cran.r-project.org/web/packages/colorspace/.

Jann, B. 2018. Customizing Stata graphs made easy (part 2). *Stata Journal* 18: 786–802.

Juul, S. 2003. Lean mainstream schemes for Stata 8 graphics. *Stata Journal* 3: 295–301.

Lin, S., J. Fortuna, C. Kulkarni, M. Stone, and J. Heer. 2013. Selecting semantically-resonant colors for data visualization. *Computer Graphics Forum* 32: 401–410.

Morris, T. 2013. scheme-mrc: Stata module to provide graphics scheme for UK Medical Research Council. Statistical Software Components S457703, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s457703.html.

———. 2015. scheme-tfl: Stata module to provide graph scheme, based on Transport for London's corporate colour pallette. Statistical Software Components S458103, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458103.html.

Okabe, M., and K. Ito. 2002. Color universal design (CUD). How to make figures and presentations that are friendly to Colorblind people. http://jfly.iam.u-tokyo.ac.jp/color/.

Pisati, M. 2007. spmap: Stata module to visualize spatial data. Statistical Software Components S456812, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s456812.html.

Tol, P. 2018. Colour schemes. Technical Note SRON/EPS/TN/09-002, SRON. https://personal.sron.nl/~pault/data/colourschemes.pdf.

Zeileis, A., K. Hornik, and P. Murrell. 2009. Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics and Data Analysis* 53: 3259–3270.

About the author

Ben Jann is a 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 the principle investigator of TREE, a large-scale multicohort panel study in Switzerland on transitions from education to employment (http://www.tree.unibe.ch).

Appendix. Symbol palettes and line-pattern palettes

The `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 `help symbolpalette` and `help linepalette`. Figure 6 shows an overview of the available named palettes.

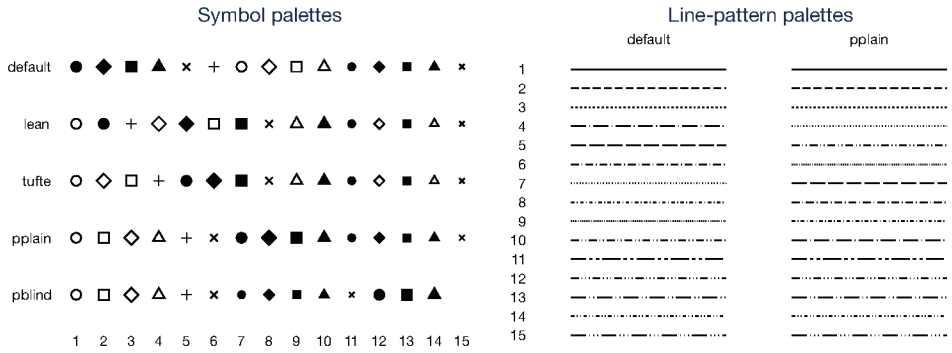


Figure 6. Symbol palettes and line-pattern palettes