colorvar_options — Options for using levels of a variable to control color

Description   Quick start   Syntax   Options   Remarks and examples
References   Also see

Description

colorvar_options allow you to create two-way plots in which the color of the markers, lines, spikes, dots, or bars varies based on the values of a numeric variable. A simple example would be when you create a scatterplot of y variable versus x variable and when you want different colored markers for levels of a third numeric variable, *colorvar*. Note that *colorvar_options* also apply to two-way plots with multiple y and x axes. These options provide fine control of the number and value of the levels formed by *colorvar*, as well as the colors used to represent them in your plots.

Quick start

Scatterplot of `y` versus `x1` with different colored markers for levels of `x2`, with levels automatically determined

```
twoway scatter y x1, colorvar(x2)
```

Same as above, but specify 6 levels for `x2`

```
twoway scatter y x1, colorvar(x2) colorlevels(6)
```

Specify cutpoints for `x2`

```
twoway scatter y x1, colorvar(x2) colorcuts(10(10)70)
```

Scatterplot of `y` versus `x` with different colored markers for each value of `a`

```
twoway scatter y x, colorvar(a) colordiscrete
```

Bar graph with different colored bars for levels of `x2`, with levels automatically determined

```
twoway bar y x1, colorvar(x2)
```

Same as above, but modify the opacity and intensity of the colors used

```
twoway bar y x, colorvar(x2) colorrule(%40 *2)
```

Use equally spaced intensities, rather than different colors, for the levels

```
twoway bar y x, colorvar(x2) colorrule(intensity)
```

Syntax

colorvar_options	Description
`colorvar(`*colorvar*`)`	specify variable to control color
`colordiscrete`	treat *colorvar* as discrete instead of continuous
`colorcuts(`*numlist*`)`	specify list of cuts for *colorvar*
`colorlevels(#)`	specify number of levels for *colorvar*
`colorvarminmax`	include minimum and maximum of *colorvar* in the list of cuts created by `colorcuts()` and `colorlevels()`
`colorrule(`[*crule*] [`%#` [`*#`]]`)`	specify rule for creating colors for levels of *colorvar*
`colorstart(`*colorstyle*`)`	specify starting color for `colorrule()`
`colorend(`*colorstyle*`)`	specify ending color for `colorrule()`
`colorlist(`*colorstylelist*`)`	specify list of colors for each level of *colorvar*
`colorfillonly`	specify that the color for items based on *colorvar* not affect the outline
`colorformissing(`*colorstyle*`)`	specify the color for the level of missing values
`colorkeysrange`	draw keys in legend as ranged bars
`coloruseplegend`	use the contour-line plot legend instead of the contour plot legend

crule	Description
`hue`	use equally spaced hues between `colorstart()` and `colorend()`; the default
`chue`	use equally spaced hues between `colorstart()` and `colorend()`; unlike `hue`, it uses 360 + `hue` of the `colorend()` if the hue of the `colorend()` is less than the hue of the `colorstart()`
`phue`	use the colors defined in pstyle `p1`–`p15`; for additional plots, `colorrule(hue)` is used
`intensity`	use equally spaced intensities with `colorend()` as the base; `colorstart()` is ignored
`linear`	use equally spaced interpolations of the RGB values between `colorstart()` and `colorend()`

Options

`colorvar(`*colorvar*`)` specifies that the color of the markers, lines, bars, spikes, or dots in a plot be determined by the values of *colorvar*, which must be numeric. The range of values for *colorvar* will be divided into levels, formed by equally spaced values, and each level will be represented by a different color. The number of levels is determined by the major ticks on the z axis. Additionally, if *colorvar* contains missing values, these missing values will form a separate category and will be included in the graph in the color gray.

When `colorvar()` is specified with `colordiscrete`, `colorcuts()`, or `colorlevels()`, these options will determine the number of levels.

Furthermore, when `colorvar()` is specified with `colorrule()` or `colorlist()`, these options will determine whether levels are represented by different colors, hues, intensities, or interpolations of RGB values.

`colordiscrete` treats *colorvar* as a discrete variable.

`colorcuts(`*numlist*`)` and `colorlevels(#)` determine how many levels are created from *colorvar* and the values of those levels. These two options may not be combined. If neither option is specified, the values used for the major ticks on the z axis are used as cuts.

An alternative way of controlling the number and value of the levels is using the standard axis-label options available through the `zlabel()` option; see [G-3] ***axis_label_options***. However, note that `colorcuts()` and `colorlevels()` generate the exact number of cuts or levels you specify, whereas `zlabel()` generates an approximation to the number of values specified. Even when `colorcuts()` or `colorlevels()` is specified, you can further control the appearance of z-axis labels using the `zlabel()` option.

`colorcuts(`*numlist*`)` specifies the breaks for the groups to be formed by *colorvar*. The values in *numlist* $(n_1, n_2, \ldots, n_k)$ will be sorted, and the levels determined as follows: $(-\infty, n_1\,], (n_1, n_2\,], (n_2, n_3\,], \ldots, (n_{k-1}, n_k\,], (n_k, \infty)$. Any values below the minimum for *colorvar* are ignored; if all values are below the minimum, the minimum value will be used as the cut.

When `colordiscrete` is specified with `colorcuts(`*numlist*`)`, then the values in *numlist* will be the levels. Any values that are not included in these levels will be grouped with missing values and displayed in gray.

`colorlevels(#)` specifies the number of levels for *colorvar*. It creates $\# - 1$ equally spaced cuts between the minimum and maximum of *colorvar* as follows: $n_1, n_2, \ldots, n_k$, where $k = \# - 1$. The levels are determined as $(-\infty, n_1\,], (n_1, n_2\,], (n_2, n_3\,], \ldots, (n_{k-1}, n_k\,], (n_k, \infty)$.

When `colordiscrete` is specified with `colorlevels()`, then the levels will be $n_1, n_2, \ldots, n_k$. Any values that are not included in these levels will be grouped with missing values and displayed in gray. `colordiscrete` is rarely specified with `colorlevels()`.

`colorvarminmax` is a modifier of `colorcuts()` and `colorlevels()` and specifies that the minimum and maximum of *colorvar* be included in the list of cuts. The major ticks on the z axis will also be updated to include the minimum and maximum values.

`colorrule(`[*crule*] [%# [*#]]`)`, `colorstart(`*colorstyle*`)`, `colorend(`*colorstyle*`)`, and `colorlist(`*colorstylelist*`)` determine the colors that are used for each level.

`colorrule(`[*crule*] [%# [*#]]`)` specifies the rule used to set the colors for each level. *crule* may be `hue`, `chue`, `phue`, `intensity`, or `linear`. The default is `colorrule(hue)`. *crule* can be modified with opacity (%#) and intensity (*#). For example, `colorrule(hue %40 *2)` means equally spaced hues will be used to represent each level of *colorvar*; `*2` means the intensity of the colors will be darker, and `%40` sets the opacity to 40%. See [G-4] ***colorstyle*** for more information on opacity and intensity.

`colorstart(`*colorstyle*`)` specifies the starting color for the rule. See [G-4] ***colorstyle***.

`colorend(`*colorstyle*`)` specifies the ending color for the rule. See [G-4] ***colorstyle***.

`colorlist(`*colorstylelist*`)` specifies a list of *colorstyle*s for the levels. If RGB, CMYK, HSV, hexadecimal, or an intensity-adjusted (for example, `red*.3`) colorstyle is specified, it should be placed in quotes. Examples of valid `colorlist()` specifications include `colorlist(red green magenta)` and `colorlist(red "55 132 22" ".3 .9 .3 hsv" blue)`.

`colorfillonly` specifies that the color for dots, bars, and markers based on *colorvar* not affect the outline color. The outline color will be determined by the corresponding plot options, such as `mlcolor()` and `mlstyle()` for `graph twoway scatter`.

By default, `colorvar(`*colorvar*`)` specifies both the fill and outline color for the dots, bars, and markers.

`colorformissing(`*colorstyle*`)` specifies the color to be used for the level of missing values. See [G-4] ***colorstyle***. By default, the color gray is used to represent missing values.

`colorkeysrange` draws keys in the legend as ranged bars, similar to a contour plot.

`coloruseplegend` specifies that keys for *colorvar* be placed in a contour-line plot legend (see `plegend()`), instead of the default contour plot legend (see `clegend()`). You must specify `colordiscrete` with `coloruseplegend`.

Remarks and examples

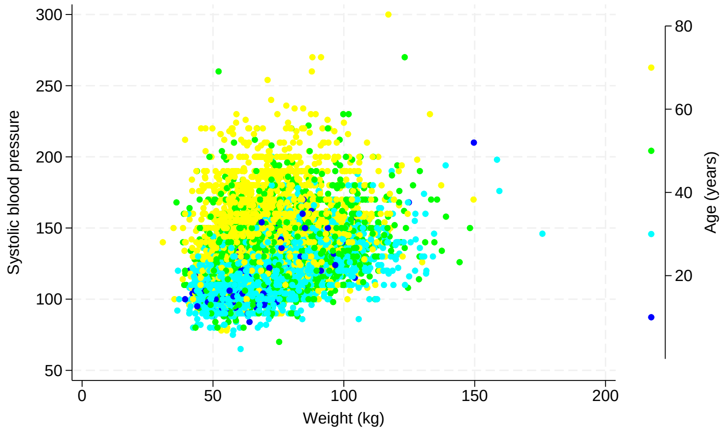
Remarks are presented under the following headings:

Introduction

When creating two-way plots, we are interested in the relationship between the numeric variables y and x. At times, it can be useful to color code these plots based on values of another variable, which we call *colorvar*; this allows us to see how the relationship between y and x differs for each level of *colorvar*. *colorvar_options* allow us to use this color coding when creating bar graphs, scatterplots, line plots, dot plots, dropped-line plots, connected plots, spike plots, time-series line plots, and several range and paired-coordinate plots.

For example, below we use data from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981). We create a scatterplot to visualize how systolic blood pressure varies with weight, and we use different colored markers for different ages:

```
. use https://www.stata-press.com/data/r19/nhanes2l
(Second National Health and Nutrition Examination Survey)
. scatter bpsystol weight, colorvar(age)
```



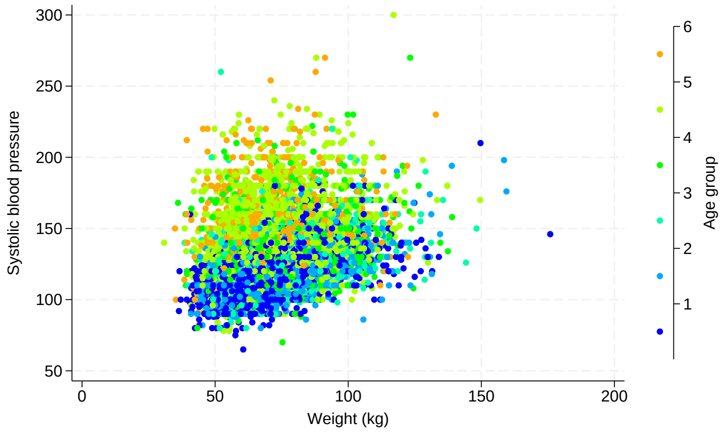
Stata automatically created four levels from the `age` variable and assigned a color to each level.

colorvar_options allow you to specify the cutpoints or the number of levels to be created for *colorvar*. Additionally, you have control over the colors used to represent each level. For example, you can specify a list of colors that different hues or intensities be used or simply modify the opacity and intensity of the colors used by default. In the following sections, we demonstrate how to control the number of levels and the colors used in the plots.

Controlling the number of levels

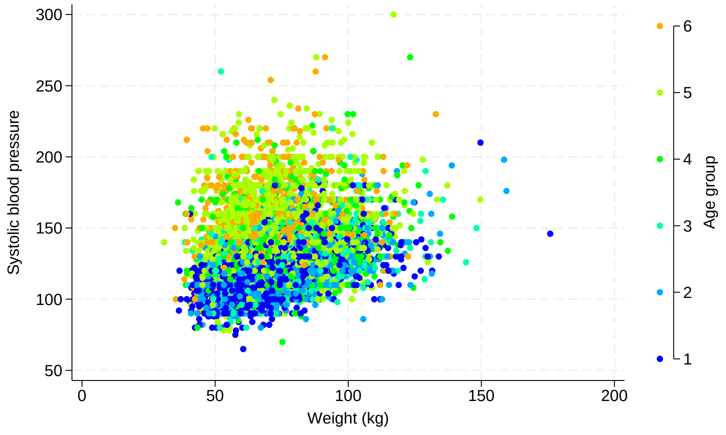
Let's modify our previous graph by using the categorical `agegrp` variable, instead of the continuous `age` variable:

```
. scatter bpsystol weight, colorvar(agegrp)
```



When you simply specify `colorvar()`, the range of values for *colorvar* will be divided into levels, formed by equally spaced values, and each level will be represented by a different color. Furthermore, the levels are determined from the major ticks on the z axis. In this particular graph, six levels were created, but the number of levels depends on the data and thus may vary. In our case, `agegrp` is a discrete variable, so we now add the `colordiscrete` option:

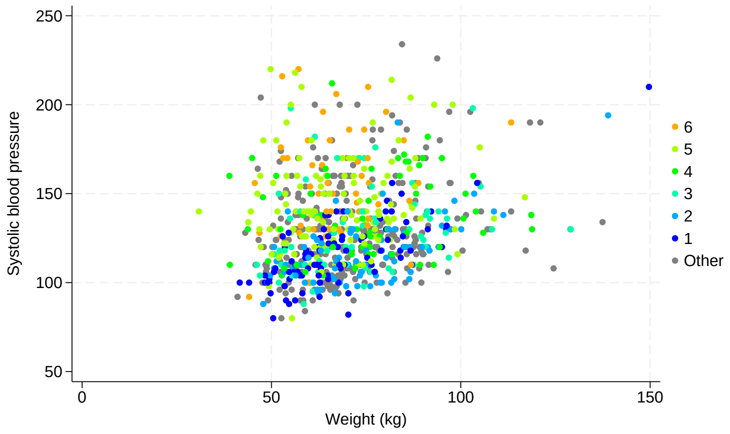
```
. scatter bpsystol weight, colorvar(agegrp) colordiscrete
```



We now see the markers lined up with the ticks in the legend. Each marker now represents one distinct value, rather than a level or group of values. The variable agegrp has six distinct values, so we get six different colors. If there were any observations with missing values for agegrp, they would show up as gray markers. Missing values of *colorvar* will form a separate level and will be included in the graph in the color gray. For example, below we replace some values of agegrp with missing values and re-create our graph using the contour-line plot legend instead of the default contour plot legend:

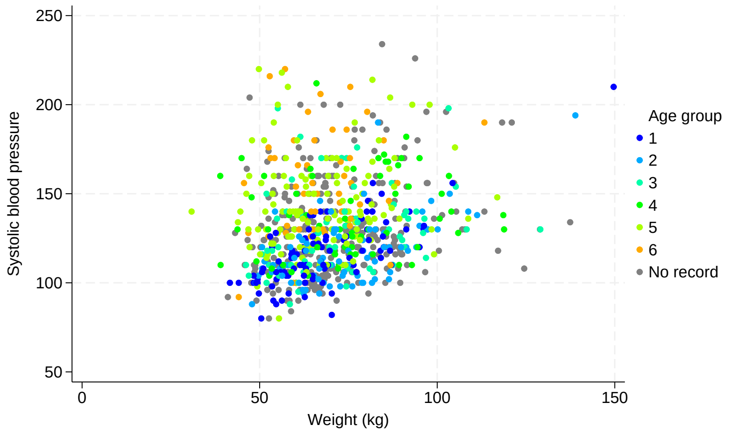
```
. replace agegrp = . in 1/300
(300 real changes made, 300 to missing)

. scatter bpsystol weight in 1/800, colorvar(agegrp) colordiscrete
> coloruseplegend
```



Here we see keys for each distinct value of age group and an "Other" category, which refers to the group of missing values. We can modify the contents of this legend using the plegend() option. Below, we reorder the keys, from the first to sixth age group. We also add the text "Age group", and we change the label for the missing group.

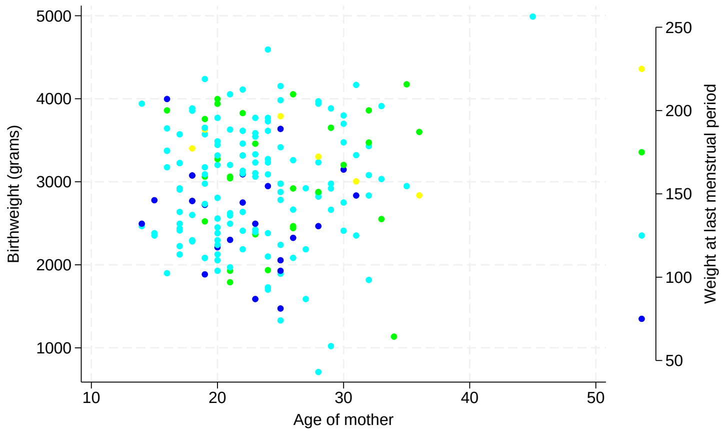
```
. scatter bpsystol weight in 1/800, colorvar(agegrp) colordiscrete
> coloruseplegend plegend(order(- "Age group" 6 5 4 3 2 1 7 "No record"))
```



As we saw above, specifying a discrete variable with `colorvar()` is straightforward. However, for continuous variables and categorical variables with an excessive number of levels, there is more flexibility in how the groups are formed.

For this example, we now load the dataset on infant birthweights described in Hosmer, Lemeshow, and Sturdivant (2013, 24). We are interested in how infant birthweights vary with the mother's age. We request different colored markers for each subset of the mother's weight:

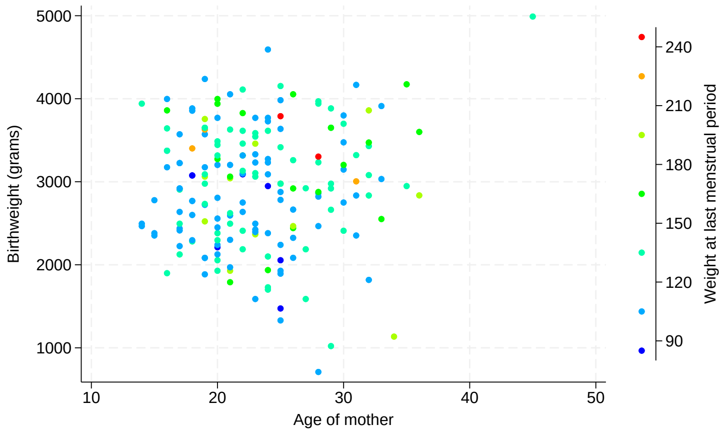
```
. use https://www.stata-press.com/data/r19/lbw, clear
(Hosmer & Lemeshow data)
. scatter bwt age, colorvar(lwt)
```



We obtain four categories for the mothers's weight: `lwt` $\leq$ 100, 100 $<$ `lwt` $\leq$ 150, 150 $<$ `lwt` $\leq$ 200, and 200 $<$ `lwt`.

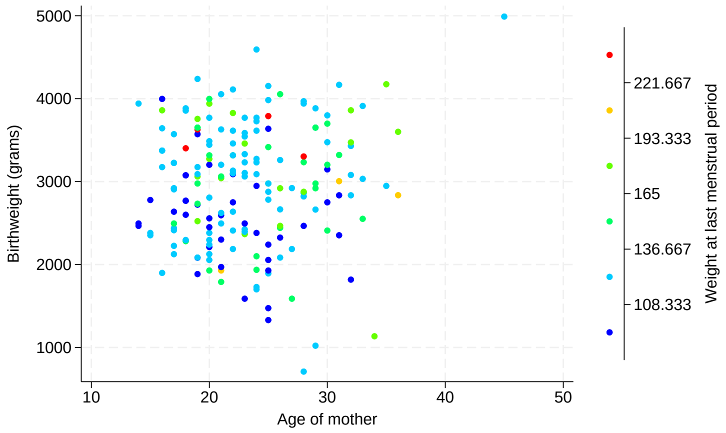
An easy way to modify the levels is to specify the cutpoints; below, we specify the cutpoints from 90 to 250, with increments of 30. Therefore, weights are grouped into the following categories: `lwt` $\leq$ 90, 90 $<$ `lwt` $\leq$ 120, 120 $<$ `lwt` $\leq$ 150, ..., 240 $<$ `lwt`.

```
. scatter bwt age, colorvar(lwt) colorcuts(90(30)250)
```



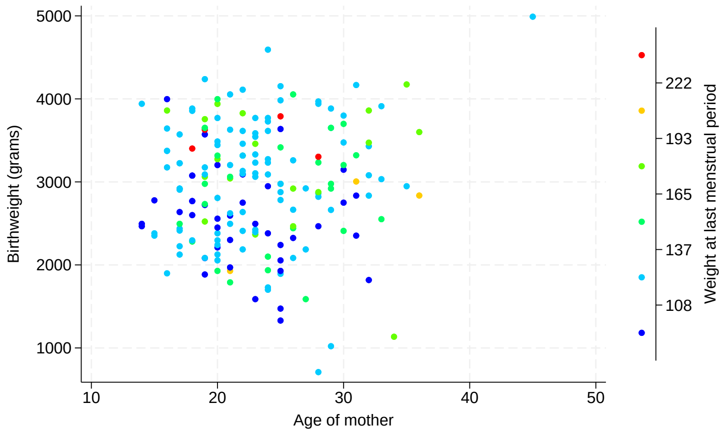
The other way we can control how the levels are formed is to specify the number of levels we want. For example, below we request six levels for `lwt`:

```
. scatter bwt age, colorvar(lwt) colorlevels(6)
```



`lwt` ranges from 80 to 250, and this range is split into 6 equally spaced groups. We get `lwt` $\leq$ 108.33, 108.33 $<$ `lwt` $\leq$ 136.67, 136.67 $<$ `lwt` $\leq$ 165, …, 221.67 $<$ `lwt`. We can modify the labels on the axis with `zlabel()`. For example, the mother's weight is recorded as an integer, so we format the labels with zero decimal places:

```
. scatter bwt age, colorvar(lwt) colorlevels(6) zlabel(, format(%6.0f))
```



When `zlabel()` is specified with `colorcuts()` or `colorlevels()`, it simply affects the labels on the z axis. When neither of options `colorcuts()` or `colorlevels()` is specified, `zlabel()` also controls the number and value of levels for *colorvar*.

However, unlike `colorcuts()` and `colorlevels()`, `zlabel()` will not necessarily create the exact number of levels we request but rather an approximation to that number. For example, if you issue the command below, you will not get six levels:

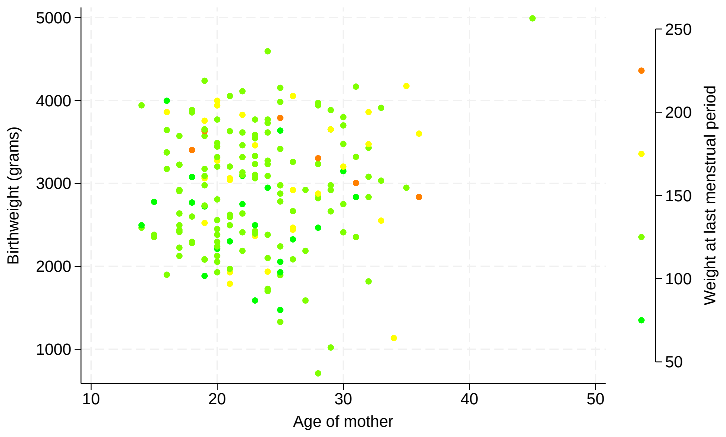
```
scatter bwt age, colorvar(lwt) zlabel(#6)
```

Controlling the colors

With `colorvar()`, we have control of the colors that are used. We can specify the color for each level, or we can have different levels represented by different hues, intensities, or interpolations of RGB values.

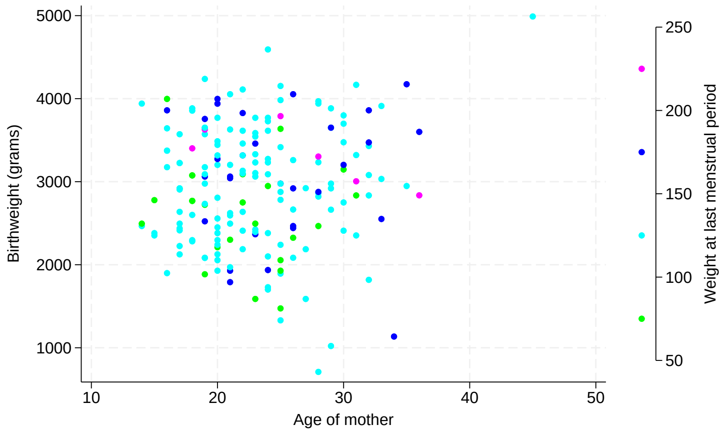
For example, when we created our first scatterplot above with levels defined by `lwt`, we saw blue, green, and yellow markers. By default, `colorvar()` uses equally spaced hues (`colorrule(hue)`) to represent the different levels. We can extend that to specify the starting and ending colors as follows:

```
. scatter bwt age, colorvar(lwt) colorstart(green) colorend(red)
```



Because the hue of green is 120 and the hue of red is 0, the hues of the levels are decreasing. So we see yellow but not blue and purple. If we instead want the hues of the levels to be increasing, we can use `colorrule(chue)`:

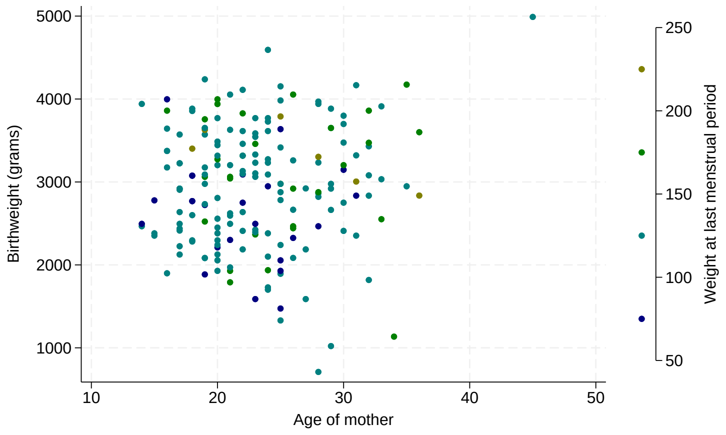
```
. scatter bwt age, colorvar(lwt) colorstart(green) colorend(red) colorrule(chue)
```



The hue of the ending color is less than the hue of the starting color, so the hue for the ending color was incremented by 360. So we get increasing hues from 120 to 360; we see blue and purple but not yellow.

We can also change the intensity and opacity of each color with `colorrule()`. For example, we can increase the intensity of the colors used by default as follows:

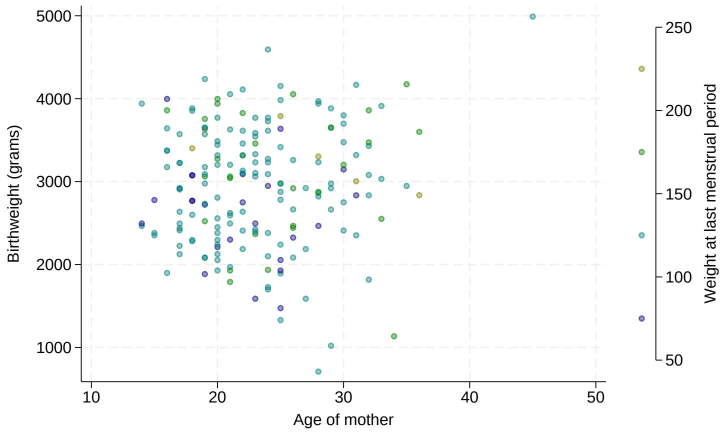
```
. scatter bwt age, colorvar(lwt) colorrule(*2)
```



Because we did not specify *crule* with `colorrule()`, the default rule `hue` was used.

We can also adjust the opacity of these darker markers; below, we set the opacity to 40%.

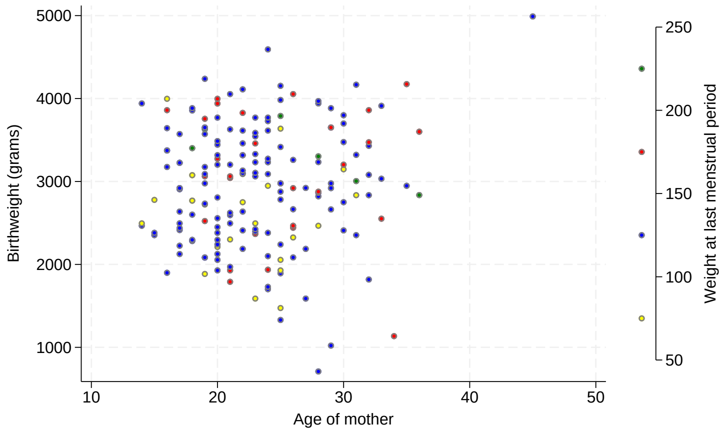
```
. scatter bwt age, colorvar(lwt) colorrule(%40 *2)
```



Note that `colorvar()` sets the color of both the fill and outline of the markers, dots, and bars of the graph. If you want to use these colors only for the fill and not the outline, you can use `colorfillonly`. You can then use the plot-specific options to modify the color of the outline.

For example, for scatterplots we can use the `mlcolor()` option to specify the color and opacity of the outline. Below, we use `colorlist()` to specify the color for each level and `mlcolor(gray)` for a gray outline:

```
. scatter bwt age, colorvar(lwt) colorlist(yellow blue red green)
> mlcolor(gray) colorfillonly
```



References

Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.

McDowell, A., A. Engel, J. T. Massey, and K. Maurer. 1981. “Plan and operation of the Second National Health and Nutrition Examination Survey, 1976–1980”. In *Vital and Health Statistics*, ser. 1, no. 15. Hyattsville, MD: National Center for Health Statistics.

Also see

[G-3] ***axis_label_options*** — Options for specifying axis labels

[G-4] ***colorstyle*** — Choices for color