graph twoway heatmap — Two-way heat map

Description

`twoway heatmap` displays values of z across values of y and x as a grid of colored rectangles. You can use a different color for each unique value of z or for different levels of z. If you have multiple z-values for the same x, y pair, the mean of the z-values will be used in the plot.

Quick start

Specify a heat map displaying `z` for rectangles defined by each distinct (`x`, `y`) pair

```
twoway heatmap z y x
```

Same as above, but specify 5 levels for `z`

```
twoway heatmap z y x, levels(5)
```

Specify binning of `y` and `x` variables into 10 levels

```
twoway heatmap z y x, ylevels(10) xlevels(10)
```

Specify grid cutpoints for `y` and `x`

```
twoway heatmap z y x, ycuts(100(10)200) xcuts(100(20)200)
```

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

```
twoway heatmap z y x, crule(intensity)
```

Same as above, but specify navy as the ending, or darkest, color

```
twoway heatmap z y x, crule(intensity) ecolor(navy)
```

Menu

Graphics > Two-way graph (scatter, line, etc.)

Syntax

```
twoway heatmap z y x [if] [in] [, options]
```

options	Description
`ccuts(`*numlist*`)`	list of values for z-variable cuts
`levels(#)`	number of levels for z variable
`minmax`	include minimum and maximum of z in levels
`ycuts(`*numlist*`)`	list of values for y-variable cuts
`ylevels(#)`	number of levels for y variable
`xcuts(`*numlist*`)`	list of values for x-variable cuts
`xlevels(#)`	number of levels for x variable
`crule(`*crule*`)`	rule for creating colors for each level of the z variable
`scolor(`*colorstyle*`)`	starting color for `crule()`
`ecolor(`*colorstyle*`)`	ending color for `crule()`
`ccolors(`*colorstylelist*`)`	list of colors for z-variable levels
twoway_options	titles, legends, axes, added lines and text, by, regions, name, aspect ratio, etc.

crule	Description
`hue`	use equally spaced hues between `scolor()` and `ecolor()`; the default
`chue`	use equally spaced hues between `scolor()` and `ecolor()`; unlike `hue`, it uses 360 + `hue` of the `ecolor()` if the hue of the `ecolor()` is less than the hue of the `scolor()`
`intensity`	use equally spaced intensities with `ecolor()` as the base; `scolor()` is ignored
`linear`	use equally spaced interpolations of the RGB values between `scolor()` and `ecolor()`

Options

`ccuts()`, `levels()`, and `minmax` determine how many levels are created from z and the values of those levels.

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***. Even when `ccuts()` or `levels()` is specified, you can further control the appearance of the z-axis labels using the `zlabel()` option.

`ccuts(`*numlist*`)` specifies the cutpoints for z used to create levels.

`levels(#)` specifies the number of levels to create for z; $\# - 1$ cuts will be created.

`minmax` is a modifier of `levels()` and specifies that the minimum and maximum values of z be included in the cuts.

`ccuts()` and `levels()` are different ways of specifying the z-variable cuts and may not be combined.

`ycuts()`, `ylevels()`, `xcuts()`, and `xlevels()` determine how observations are binned together by the y and x variables. Below, we describe how `xcuts()` and `xlevels()` work, but the same logic applies to `ycuts()` and `ylevels()`.

Given a sequence of distinct values for x,

$$x_1 < x_2 < \cdots < x_{n-1}$$

data are divided into n bins as

$$(-\infty, x_1), (x_1, x_2], \ldots (x_{n-1}, \infty)$$

However, instead of $-\infty$ and ∞, we use the minimum and maximum values *xmin* and *xmax* as boundaries for the grid. If $x_1 = xmin$, then the lower boundary of the grid is expanded to $x_1 -$ step, where step $= (x_2 - x_1)/2$.

If you do not specify any of the `ycuts()`, `ylevels()`, `xcuts()`, or `xlevels()` options, each distinct pair of (`y`, `x`) is placed within one rectangle.

Suppose the distinct values of the x variable are

$$x_1 < x_2 < \cdots < x_n$$

where x_1 is the minimum and x_n is the maximum. The cutpoints that define the grid are

$$xc_0 < xc_1 < xc_2 < \cdots < xc_{n-1} < xc_n$$

where $xc_i = (x_i + x_{i+1})/2$ for $xc_1 \cdots xc_{n-1}$. Additionally, $xc_0 = x_1 - (x_2 - x_1)/2$ and $xc_n = x_n + (x_n - x_{n-1})/2$.

For `xlevels(`*n*`)` cutpoints are defined as

$$xc_1 < xc_2 < \cdots < xc_{n-1}$$

where $xc_i = xmin + i * (xmax - xmin)/n$.

For `xcuts(`*numlist*`)`, cutpoints are defined as

$$x_1 < x_2 < \cdots < x_{n-1}$$

where x_i are all the unique values contained in *numlist* within $[xmin, xmax]$ and sorted in ascending order. Any values x_i outside $[xmin, xmax]$ are ignored.

`crule()`, `scolor()`, `ecolor()`, and `ccolors()` determine the colors that are used to fill the rectangles for each level of z.

`crule(`*crule*`)` specifies the rule used to set the colors for the levels of z. Valid *crule*s are `hue`, `chue`, `intensity`, and `linear`. The default is `crule(hue)`.

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

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

`ccolors(`*colorstylelist*`)` specifies a list of *colorstyle*s for the rectangles of each level of z. 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 `ccolors()` options include `ccolors(red green magenta)` and `ccolors(red "55 132 22" ".3 .9 .3 hsv" blue)`. See [G-4] ***colorstyle***.

twoway_options are any of the options documented in [G-3] ***twoway_options***. These include options for titling the graph (see [G-3] ***title_options***); for saving the graph to disk (see [G-3] ***saving_option***); for controlling the labeling and look of the axes (see [G-3] ***axis_options***); for controlling the look, contents, position, and organization of the legend (see [G-3] ***legend_options***); for adding lines (see [G-3] ***added_line_options***) and text (see [G-3] ***added_text_options***); and for controlling other aspects of the graph's appearance (see [G-3] ***twoway_options***).

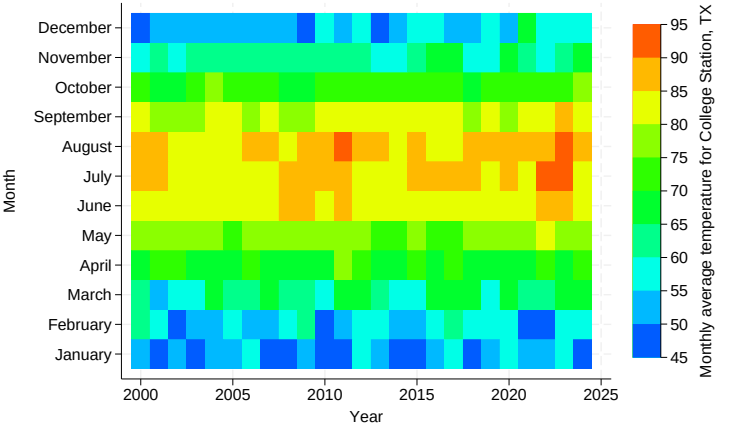
Remarks and examples

Remarks are presented under the following headings:

Introduction

`twoway heatmap` displays values of z across values of y and x as a grid of colored rectangles. For example, we have the monthly average temperature recorded for the city of College Station, Texas.

```
. use https://www.stata-press.com/data/r19/cstemp
(Monthly average temperature for College Station, Texas)
. twoway heatmap temp month year, ylabel(1(1)12, valuelabel) ccuts(45(5)95)
```



We have a rectangle for each month and year, and we fill in each rectangle with a color based on the temperature. We specified the cutpoints for the levels of temperature, from 45 to 95, in increments of 5; this gives us different-colored rectangles for these different levels of temperature.

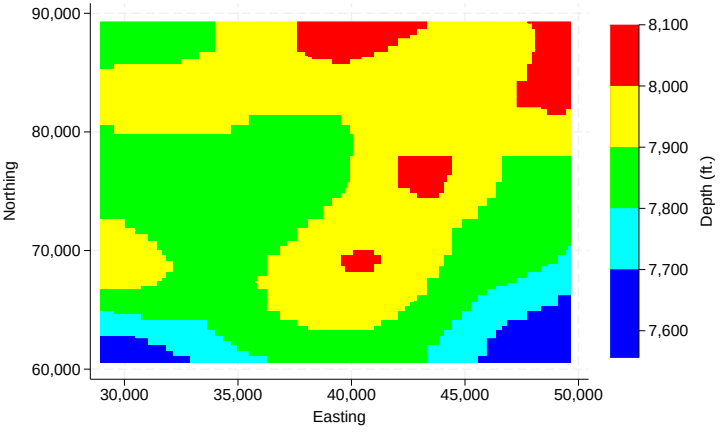
In this example, we had one value of z, temperature, for each (x, y) pair, year and month. If your data have more than one z-value corresponding to an (x, y) pair, the mean of the z-values is used in the plot. For example, below we have two values of z for $(1, 4)$, so the mean value, 2.5, would be used in the heat map.

```
. input z y x
      z x y
1.  1 1 1
2.  2 1 4
3.  3 1 4
4.  1 2 1
5.  1 2 4
6.  end
```

Controlling the number of bins for y and x

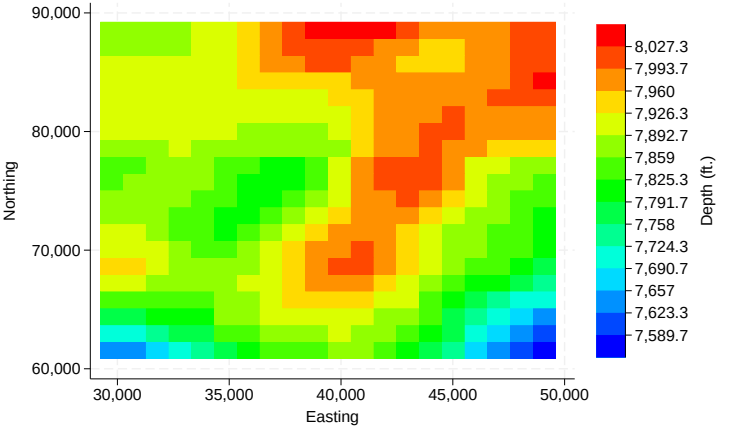
We have artificial data on sandstone elevation in Ohio. We could draw a heat map with default values by typing

```
. use https://www.stata-press.com/data/r19/sandstone
(Subsea elevation of Lamont sandstone in an area of Ohio)
. twoway heatmap depth northing easting
```



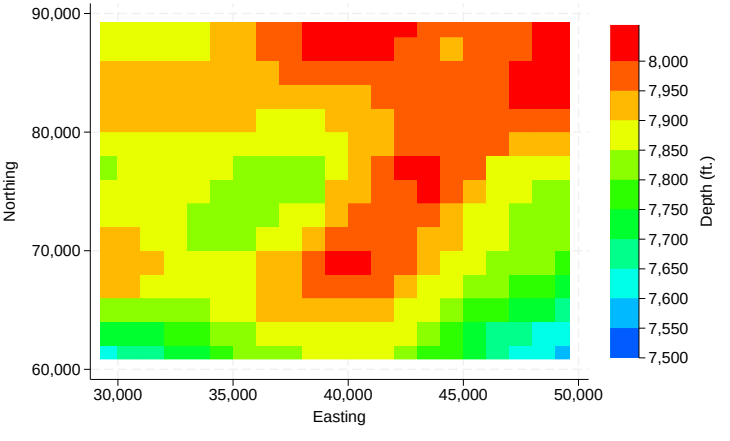
We can use the `xlevels()` and `ylevels()` options to create bins of equally divided intervals along the x and y axes. Additionally, we want to categorize `depth` into 15 levels:

```
. twoway heatmap depth northing easting, xlevels(20) ylevels(20) levels(15)
```



We can roughly see that we have a 20×20 grid. If we want to explicitly specify the cutpoints for the levels of x and y, we can use the `xcuts()` and `ycuts()` options. Similarly, we use `ccuts()` to specify integers as the cutpoints for the values of `depth`.

```
. twoway heatmap depth northing easting, xcuts(30000(1000)50000)
        ycuts(60000(2000)90000) ccuts(7500(50)8000)
```



Also see

[G-2] **graph twoway contour** — Two-way contour plot with area shading

[G-2] **graph twoway contourline** — Two-way contour-line plot

[G-2] **graph twoway area** — Two-way line plot with area shading

[G-2] **graph twoway rarea** — Range plot with area shading

For suggested citations, see the FAQ on citing Stata documentation.