 [Chart.js](http://docs.google.com/docs/3.9.1/)

[Home](http://docs.google.com/docs/3.9.1/)

[API](http://docs.google.com/docs/3.9.1/api/)

[Samples](http://docs.google.com/docs/3.9.1/samples/)

Ecosystem Ecosystem

* [Awesome (opens new window)](https://github.com/chartjs/awesome)
* [Slack (opens new window)](https://chartjs-slack.herokuapp.com/)
* [Stack Overflow (opens new window)](https://stackoverflow.com/questions/tagged/chart.js)

[GitHub (opens new window)](https://github.com/chartjs/Chart.js)

[Home](http://docs.google.com/docs/3.9.1/)

[API](http://docs.google.com/docs/3.9.1/api/)

[Samples](http://docs.google.com/docs/3.9.1/samples/)

Ecosystem Ecosystem

* [Awesome (opens new window)](https://github.com/chartjs/awesome)
* [Slack (opens new window)](https://chartjs-slack.herokuapp.com/)
* [Stack Overflow (opens new window)](https://stackoverflow.com/questions/tagged/chart.js)

[GitHub (opens new window)](https://github.com/chartjs/Chart.js)

* [Information](http://docs.google.com/docs/3.9.1/samples/information.html)
* Bar Charts
* Line Charts
* Other charts
* Area charts
* Scales
* Scale Options
* Legend
* Title
* Subtitle
* Tooltip
* Scriptable Options
  + [Bar Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/bar.html)
  + [Bubble Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/bubble.html)
  + [Line Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/line.html)
  + [Pie Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/pie.html)
  + [Polar Area Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/polar.html)
  + [Radar Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/radar.html)
* Animations
* Advanced
* Plugins
* [Utils](http://docs.google.com/docs/3.9.1/samples/utils.html)

[**#**](#gjdgxs) Bubble Chart

options data setup

function channelValue(x, y, values) { return x < 0 && y < 0 ? values[0] : x < 0 ? values[1] : y < 0 ? values[2] : values[3]; } function colorize(opaque, context) { const value = context.raw; const x = value.x / 100; const y = value.y / 100; const r = channelValue(x, y, [250, 150, 50, 0]); const g = channelValue(x, y, [0, 50, 150, 250]); const b = channelValue(x, y, [0, 150, 150, 250]); const a = opaque ? 1 : 0.5 \* value.v / 1000; return 'rgba(' + r + ',' + g + ',' + b + ',' + a + ')'; } const config = { type: 'bubble', data: data, options: { aspectRatio: 1, plugins: { legend: false, tooltip: false, }, elements: { point: { backgroundColor: colorize.bind(null, false), borderColor: colorize.bind(null, true), borderWidth: function(context) { return Math.min(Math.max(1, context.datasetIndex + 1), 8); }, hoverBackgroundColor: 'transparent', hoverBorderColor: function(context) { return Utils.color(context.datasetIndex); }, hoverBorderWidth: function(context) { return Math.round(8 \* context.raw.v / 1000); }, radius: function(context) { const size = context.chart.width; const base = Math.abs(context.raw.v) / 1000; return (size / 24) \* base; } } } } };

function channelValue(x, y, values) {  
 return x < 0 && y < 0 ? values[0] : x < 0 ? values[1] : y < 0 ? values[2] : values[3];  
}  
function colorize(opaque, context) {  
 const value = context.raw;  
 const x = value.x / 100;  
 const y = value.y / 100;  
 const r = channelValue(x, y, [250, 150, 50, 0]);  
 const g = channelValue(x, y, [0, 50, 150, 250]);  
 const b = channelValue(x, y, [0, 150, 150, 250]);  
 const a = opaque ? 1 : 0.5 \* value.v / 1000;  
 return 'rgba(' + r + ',' + g + ',' + b + ',' + a + ')';  
}  
const config = {  
 type: 'bubble',  
 data: data,  
 options: {  
 aspectRatio: 1,  
 plugins: {  
 legend: false,  
 tooltip: false,  
 },  
 elements: {  
 point: {  
 backgroundColor: colorize.bind(null, false),  
 borderColor: colorize.bind(null, true),  
 borderWidth: function(context) {  
 return Math.min(Math.max(1, context.datasetIndex + 1), 8);  
 },  
 hoverBackgroundColor: 'transparent',  
 hoverBorderColor: function(context) {  
 return Utils.color(context.datasetIndex);  
 },  
 hoverBorderWidth: function(context) {  
 return Math.round(8 \* context.raw.v / 1000);  
 },  
 radius: function(context) {  
 const size = context.chart.width;  
 const base = Math.abs(context.raw.v) / 1000;  
 return (size / 24) \* base;  
 }  
 }  
 }  
 }  
};

function generateData() { const data = []; let i; for (i = 0; i < DATA\_COUNT; ++i) { data.push({ x: Utils.rand(MIN\_XY, MAX\_XY), y: Utils.rand(MIN\_XY, MAX\_XY), v: Utils.rand(0, 1000) }); } return data; } const data = { datasets: [{ data: generateData() }, { data: generateData() }] };

function generateData() {  
 const data = [];  
 let i;  
 for (i = 0; i < DATA\_COUNT; ++i) {  
 data.push({  
 x: Utils.rand(MIN\_XY, MAX\_XY),  
 y: Utils.rand(MIN\_XY, MAX\_XY),  
 v: Utils.rand(0, 1000)  
 });  
 }  
 return data;  
}  
const data = {  
 datasets: [{  
 data: generateData()  
 }, {  
 data: generateData()  
 }]  
};

const DATA\_COUNT = 16; const MIN\_XY = -150; const MAX\_XY = 100; Utils.srand(110); const actions = [ { name: 'Randomize', handler(chart) { chart.data.datasets.forEach(dataset => { dataset.data = generateData(); }); chart.update(); } }, ];

const DATA\_COUNT = 16;  
const MIN\_XY = -150;  
const MAX\_XY = 100;  
Utils.srand(110);  
const actions = [  
 {  
 name: 'Randomize',  
 handler(chart) {  
 chart.data.datasets.forEach(dataset => {  
 dataset.data = generateData();  
 });  
 chart.update();  
 }  
 },  
];

## [**#**](#30j0zll) Docs

* [Bubble](http://docs.google.com/docs/3.9.1/charts/bubble.html)
* [Options](http://docs.google.com/docs/3.9.1/general/options.html)
  + [Scriptable Options](http://docs.google.com/docs/3.9.1/general/options.html#scriptable-options)

Last Updated: 8/3/2022, 12:46:38 PM

←  [Bar Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/bar.html)   [Line Chart](http://docs.google.com/docs/3.9.1/samples/scriptable/line.html)  →