<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>

<link type="text/css" rel="stylesheet" href="style.css"/>

<style type="text/css">

svg {

font-size:

20px;

}

.foreground path {

fill: none;

}

.foreground path.fade {

stroke: #000;

stroke-opacity: 0;

}

.legend {

font-size:0px;

font-style: oblique;

}

.legend line {

stroke-width: 0px;

stroke-opacity: 0;

}

.DI {

stroke: #000000;

stroke-opacity: 1;

stroke-width: 0.5px;

}

.control {

stroke: #FFFFFF;

stroke-opacity: 0;

stroke-width: 0px;

}

.brush .extent {

fill-opacity: .3;

stroke: #fff;

shape-rendering: crispEdges;

}

.axis line, .axis path {

fill: none;

stroke: #000;

shape-rendering: crispEdges;

}

.axis text {

text-shadow: 5 0px 0 #000;

cursor: move;

}

</style>

</head>

<body>

<script type="text/javascript" src="d3.js"></script>

<script type="text/javascript" src="d3.csv.js"></script>

<script type="text/javascript">

var species = ["DI", "control"],

traits = ["PB2\_5", "PB2\_3"];

var m = [60, 30, 200, 2],

w = 600 - m[0] - m[3],

h = 800 - m[0] - m[2];

var x = d3.scale.ordinal().domain(traits).rangePoints([300, w]),

y = {};

var line = d3.svg.line(),

axis = d3.svg.axis().orient("left"),

foreground;

var svg = d3.select("body").append("svg:svg")

.attr("width", w + m[1] + m[3])

.attr("height", h + m[0] + m[2])

.append("svg:g")

.attr("transform", "translate(" + m[3] + "," + m[0] + ")");

d3.csv("PR8\_PB2\_DI.csv", function(flowers) {

// Create a scale and brush for each trait.

traits.forEach(function(d) {

// Coerce values to numbers.

flowers.forEach(function(p) { p[d] = +p[d]; });

y[d] = d3.scale.linear()

.domain(d3.extent(flowers, function(p) { return p[d]; }))

.range([h, 0]);

y[d].brush = d3.svg.brush()

.y(y[d])

.on("brush", brush);

});

// Add a legend.

var legend = svg.selectAll("g.legend")

.data(species)

.enter().append("svg:g")

.attr("class", "legend")

.attr("transform", function(d, i) { return "translate(0," + (i \* 20 + 584) + ")"; });

legend.append("svg:line")

.attr("class", String)

.attr("x2", 8);

legend.append("svg:text")

.attr("x", 12)

.attr("dy", ".31em")

.text(function(d) { return "Iris " + d; });

// Add foreground lines.

foreground = svg.append("svg:g")

.attr("class", "foreground")

.selectAll("path")

.data(flowers)

.enter().append("svg:path")

.attr("d", path)

.attr("class", function(d) { return d.species; });

// Add a group element for each trait.

var g = svg.selectAll(".trait")

.data(traits)

.enter().append("svg:g")

.attr("class", "trait")

.attr("transform", function(d) { return "translate(" + x(d) + ")"; })

.call(d3.behavior.drag()

.origin(function(d) { return {x: x(d)}; })

.on("dragstart", dragstart)

.on("drag", drag)

.on("dragend", dragend));

// Add an axis and title.

g.append("svg:g")

.attr("class", "axis")

.each(function(d) { d3.select(this).call(axis.scale(y[d])); })

.append("svg:text")

.attr("text-anchor", "middle")

.attr("y", -9)

.text(String);

// Add a brush for each axis.

g.append("svg:g")

.attr("class", "brush")

.each(function(d) { d3.select(this).call(y[d].brush); })

.selectAll("rect")

.attr("x", -8)

.attr("width", 16);

function dragstart(d) {

i = traits.indexOf(d);

}

function drag(d) {

x.range()[i] = d3.event.x;

traits.sort(function(a, b) { return x(a) - x(b); });

g.attr("transform", function(d) { return "translate(" + x(d) + ")"; });

foreground.attr("d", path);

}

function dragend(d) {

x.domain(traits).rangePoints([0, w]);

var t = d3.transition().duration(500);

t.selectAll(".trait").attr("transform", function(d) { return "translate(" + x(d) + ")"; });

t.selectAll(".foreground path").attr("d", path);

}

});

// Returns the path for a given data point.

function path(d) {

return line(traits.map(function(p) { return [x(p), y[p](d[p])]; }));

}

// Handles a brush event, toggling the display of foreground lines.

function brush() {

var actives = traits.filter(function(p) { return !y[p].brush.empty(); }),

extents = actives.map(function(p) { return y[p].brush.extent(); });

foreground.classed("fade", function(d) {

return !actives.every(function(p, i) {

return extents[i][0] <= d[p] && d[p] <= extents[i][1];

});

});

}

</script>

</body>

</html>