**DATA VISUALISATION**

**ASSIGNMENT**

**SURYA S S**

**19BIT053**

**PROTOVIS:**

**Definition:**

Protovis composes custom views of data with simple marks such as bars and dots. Unlike low-level graphics libraries that quickly become tedious for visualization, Protovis defines marks through dynamic properties that encode data, allowing inheritance, scales and layouts to simplify construction.

**Types of protovis:**

* Marks
* Panles
* Built in types
  + 1. Lines
    2. Area
    3. Bar
    4. Dot
    5. Wedge
    6. Image line
    7. Label and bar
    8. Rule and bar

1.CODE:

<html>

<head>

<title>My God, it's full of eyes!</title>

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

<script type="text/javascript" src="../protovis-r3.2.js"></script>

<style type="text/css">

body {

**background:** #222;

}

**#fig** {

**width:** 200px;

**height:** 200px;

}

</style>

</head>

<body><div id="center"><div id="fig">

<script type="text/javascript+protovis">

**var** vis **=** **new** pv.Panel()

.width(200)

.height(200)

.fillStyle("#666")

.strokeStyle("#ccc");

vis.add(pv.Panel)

.data([{x**:**50, y**:**16, r**:**40},

{x**:**64, y**:**85, r**:**20},

{x**:**90, y**:**200, r**:**60},

{x**:**150, y**:**44, r**:**20},

{x**:**175, y**:**120, r**:**40}])

.left(**function**(d) d.x)

.top(**function**(d) d.y)

.add(pv.Dot)

.fillStyle("#fff")

.strokeStyle(**null**)

.size(**function**(d) d.r **\*** d.r)

.add(pv.Dot)

.def("v", **function**(d) {

**var** m **=** **this**.mouse();

**return** (m.length() **>** d.r **/** 2) **?** m.norm().times(d.r **/** 2) **:** m;

})

.fillStyle("#aaa")

.left(**function**(d) **this**.v().x)

.top(**function**(d) **this**.v().y)

.size(**function**(d) d.r **\*** d.r **/** 4);

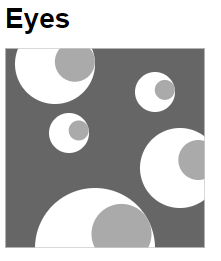
vis.render();

pv.listen(self, "mousemove", **function**() vis.render());

</script>

</div></div></body>

</html>



2.

<html>

<head>

<title>Matrix Diagram</title>

<link type="text/css" rel="stylesheet" href="ex.css?3.2"/>

<script type="text/javascript" src="../protovis-r3.2.js"></script>

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

<style type="text/css">

**#fig** {

**width:** 800px;

**height:** 800px;

}

</style>

</head>

<body><div id="center"><div id="fig">

<script type="text/javascript+protovis">

**var** color **=** pv.Colors.category19().by(**function**(d) d.group);

**var** vis **=** **new** pv.Panel()

.width(693)

.height(693)

.top(90)

.left(90);

**var** layout **=** vis.add(pv.Layout.Matrix)

.nodes(miserables.nodes)

.links(miserables.links)

.sort(**function**(a, b) b.group **-** a.group);

layout.link.add(pv.Bar)

.fillStyle(**function**(l) l.linkValue

**?** ((l.targetNode.group **==** l.sourceNode.group)

**?** color(l.sourceNode) **:** "#555") **:** "#eee")

.antialias(**false**)

.lineWidth(1);

layout.label.add(pv.Label)

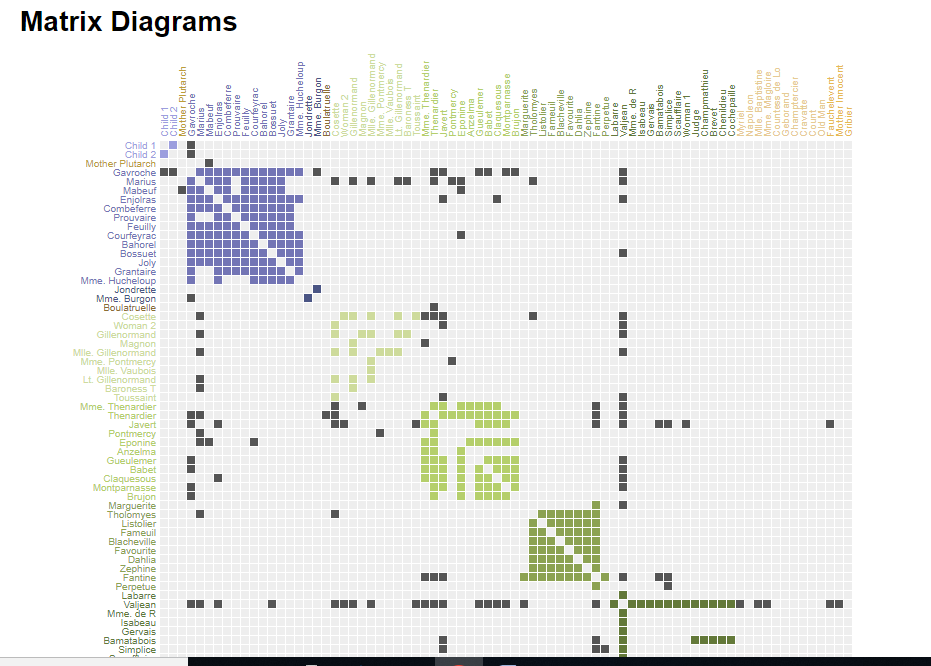
.textStyle(color);

vis.render();

</script>

</div></div></body>

</html>



3.

<html>

<head>

<title>Parallel Coordinates</title>

<link type="text/css" rel="stylesheet" href="ex.css?3.2"/>

<script type="text/javascript" src="../protovis-r3.2.js"></script>

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

<style type="text/css">

**#fig** {

**width:** 880px;

**height:** 460px;

}

**#title** {

**position:** **absolute**;

**top:** 70px;

**left:** 200px;

**padding:** 10px;

**background:** white;

}

large {

**font-size:** **medium**;

}

</style>

</head>

<body><div id="center"><div id="fig">

<script type="text/javascript+protovis">

*// The units and dimensions to visualize, in order.*

**var** units **=** {

cyl**:** {name**:** "cylinders", unit**:** ""},

dsp**:** {name**:** "displacement", unit**:** " sq in"},

lbs**:** {name**:** "weight", unit**:** " lbs"},

hp**:** {name**:** "horsepower", unit**:** " hp"},

acc**:** {name**:** "acceleration (0-60 mph)", unit**:** " sec"},

mpg**:** {name**:** "mileage", unit**:** " mpg"},

year**:** {name**:** "year", unit**:** ""}

}

**var** dims **=** pv.keys(units);

*/\* Sizing and scales. \*/*

**var** w **=** 820,

h **=** 420,

fudge **=** 0.5,

x **=** pv.Scale.ordinal(dims).splitFlush(0, w),

y **=** pv.dict(dims, **function**(t) pv.Scale.linear(

cars.filter(**function**(d) **!**isNaN(d[t])),

**function**(d) Math.floor(d[t])**-**fudge,

**function**(d) Math.ceil(d[t]) **+**fudge

).range(0, h)),

c **=** pv.dict(dims, **function**(t) pv.Scale.linear(

cars.filter(**function**(d) **!**isNaN(d[t])),

**function**(d) Math.floor(d[t])**-**fudge,

**function**(d) Math.ceil(d[t]) **+**fudge

).range("steelblue", "brown"));

*/\* Interaction state. \*/*

**var** filter **=** pv.dict(dims, **function**(t) {

**return** {min**:** y[t].domain()[0], max**:** y[t].domain()[1]};

}), active **=** "mpg";

*/\* The root panel. \*/*

**var** vis **=** **new** pv.Panel()

.width(w)

.height(h)

.left(30)

.right(30)

.top(30)

.bottom(20);

*// The parallel coordinates display.*

vis.add(pv.Panel)

.data(cars)

.visible(**function**(d) dims.every(**function**(t)

(d[t] **>=** filter[t].min) **&&** (d[t] **<=** filter[t].max)))

.add(pv.Line)

.data(dims)

.left(**function**(t, d) x(t))

.bottom(**function**(t, d) y[t](d[t]))

.strokeStyle("#ddd")

.lineWidth(1)

.antialias(**false**);

*// Rule per dimension.*

rule **=** vis.add(pv.Rule)

.data(dims)

.left(x);

*// Dimension label*

rule.anchor("top").add(pv.Label)

.top(**-**12)

.font("bold 10px sans-serif")

.text(**function**(d) units[d].name);

*// The parallel coordinates display.*

**var** change **=** vis.add(pv.Panel);

**var** line **=** change.add(pv.Panel)

.data(cars)

.visible(**function**(d) dims.every(**function**(t)

(d[t] **>=** filter[t].min) **&&** (d[t] **<=** filter[t].max)))

.add(pv.Line)

.data(dims)

.left(**function**(t, d) x(t))

.bottom(**function**(t, d) y[t](d[t]))

.strokeStyle(**function**(t, d) c[active](d[active]))

.lineWidth(1);

*// Updater for slider and resizer.*

**function** update(d) {

**var** t **=** d.dim;

filter[t].min **=** Math.max(y[t].domain()[0], y[t].invert(h **-** d.y **-** d.dy));

filter[t].max **=** Math.min(y[t].domain()[1], y[t].invert(h **-** d.y));

active **=** t;

change.render();

**return** **false**;

}

*// Updater for slider and resizer.*

**function** selectAll(d) {

**if** (d.dy **<** 3) {

**var** t **=** d.dim;

filter[t].min **=** Math.max(y[t].domain()[0], y[t].invert(0));

filter[t].max **=** Math.min(y[t].domain()[1], y[t].invert(h));

d.y **=** 0; d.dy **=** h;

active **=** t;

change.render();

}

**return** **false**;

}

*/\* Handle select and drag \*/*

**var** handle **=** change.add(pv.Panel)

.data(dims.map(**function**(dim) { **return** {y**:**0, dy**:**h, dim**:**dim}; }))

.left(**function**(t) x(t.dim) **-** 30)

.width(60)

.fillStyle("rgba(0,0,0,.001)")

.cursor("crosshair")

.event("mousedown", pv.Behavior.select())

.event("select", update)

.event("selectend", selectAll)

.add(pv.Bar)

.left(25)

.top(**function**(d) d.y)

.width(10)

.height(**function**(d) d.dy)

.fillStyle(**function**(t) t.dim **==** active

**?** c[t.dim]((filter[t.dim].max **+** filter[t.dim].min) **/** 2)

**:** "hsla(0,0,50%,.5)")

.strokeStyle("white")

.cursor("move")

.event("mousedown", pv.Behavior.drag())

.event("dragstart", update)

.event("drag", update);

handle.anchor("bottom").add(pv.Label)

.textBaseline("top")

.text(**function**(d) filter[d.dim].min.toFixed(0) **+** units[d.dim].unit);

handle.anchor("top").add(pv.Label)

.textBaseline("bottom")

.text(**function**(d) filter[d.dim].max.toFixed(0) **+** units[d.dim].unit);

vis.render();

</script>

</div></div></body>

</html>

