```
<script>

var T, opt;

var Y; // tsne result stored here
var data;

function updateEmbedding() {

  // get current solution
  var Y = T.getSolution();
  // move the groups accordingly
  gs.attr("transform", function(d, i) { return "translate(" +
                                        ((Y[i][0]*20*ss + tx) + 400) + "," +
                                        ((Y[i][1]*20*ss + ty) + 400) + ")"; });

}

var svg;
function initEmbedding() {
  $("#embed").empty();
  var div = d3.select("#embed");
  svg = div.append("svg") // svg is global
    .attr("width", 1140)
    .attr("height", 1140);
}

var gs;
var cs;
var ts;
function drawEmbedding() {

    gs = svg.selectAll(".b")
      .data(data)
      .enter().append("g")
      .attr("class", "u");

    cs = gs.append("circle")
      .attr("cx", 0)
      .attr("cy", 0)
      .attr("r", 5)
      .attr('stroke-width', 1)
      .attr('stroke', 'black')
      .attr('fill', 'rgb(100,100,255)');

    if(labels.length > 0) {
      ts = gs.append("text")
        .attr("text-anchor", "top")
        .attr("transform", "translate(5, -5)")
        .attr("font-size", 12)
        .attr("fill", "#333")
        .text(function(d,i) { return labels[i]; });
    }
```

```
    var zoomListener = d3.behavior.zoom()
      .scaleExtent([0.1, 10])
      .center([0,0])
      .on("zoom", zoomHandler);
    zoomListener(svg);
}

var tx=0, ty=0;
var ss=1;
function zoomHandler() {
  tx = d3.event.translate[0];
  ty = d3.event.translate[1];
  ss = d3.event.scale;
}

var stepnum = 0;
function step() {
  if(dotrain) {
    var cost = T.step(); // do a few steps
    $("#cost").html("iteration " + T.iter + ", cost: " + cost);
  }
  updateEmbedding();
}

labels = [];
function preProLabels() {
  var txt = $("#inlabels").val();
  var lines = txt.split("\n");
  labels = [];
  for(var i=0;i<lines.length;i++) {
    var row = lines[i];
    if (! /\S/.test(row)) {
      // row is empty and only has whitespace
      continue;
    }
    labels.push(row);
  }
}

dataok = false;
function preProData() {
  var txt = $("#incsv").val();
  var d = $("#deltxt").val();
  var lines = txt.split("\n");
  var raw_data = [];
  var dlen = -1;
  dataok = true;
  for(var i=0;i<lines.length;i++) {
    var row = lines[i];
    if (! /\S/.test(row)) {
      // row is empty and only has whitespace
      continue;
```

```
    }
    var cells = row.split(d);
    var data_point = [];
    for(var j=0;j<cells.length;j++) {
      if(cells[j].length !== 0) {
        data_point.push(parseFloat(cells[j]));
      }
    }
    var dl = data_point.length;
    if(i === 0) { dlen = dl; }
    if(dlen !== dl) {
      // TROUBLE. Not all same length.
      console.log('TROUBLE: row ' + i + ' has bad length ' + dlen);
      dlen = dl; // hmmm...
      dataok = false;
    }
    raw_data.push(data_point);
  }
  data = raw_data; // set global
}

dotrain = true;
iid = -1;
$(window).load(function() {

  initEmbedding();

  $("#stopbut").click(function() {
    dotrain = false;
  });

  $("#inbut").click(function() {

    initEmbedding();
    preProData();
    if(!dataok) { // this is so terrible... globals everywhere #fasthacking #sosorry
      alert('there was trouble with data, probably rows had different number of elements. See
      console for output.');
      return;
    }
    preProLabels();
    if(labels.length > 0) {
      if(data.length !== labels.length) {
        alert('number of rows in Text labels ('+labels.length+') does not match number of rows
        in Data (' + data.length + ')! Aborting.');
        return;
      }
    }

    // ok lets do this
    opt = {epsilon: parseFloat($("#lrtxt").val()), perplexity: parseInt($("#perptxt").val()),
    dim: data[0].length};
    T = new tsnejs.tSNE(opt); // create a tSNE instance
```

```
    var dfv = $('input[name=rdata]:checked', '#datatypeform').val();
    if(dfv === 'raw') {
      console.log('raw');
      T.initDataRaw(data);
    }
    if(dfv === 'dist') {
      console.log('dist');
      T.initDataDist(data);
    }
    drawEmbedding();
    iid = setInterval(step, 10);
    dotrain = true;

  });
});
```

```
</script>
```