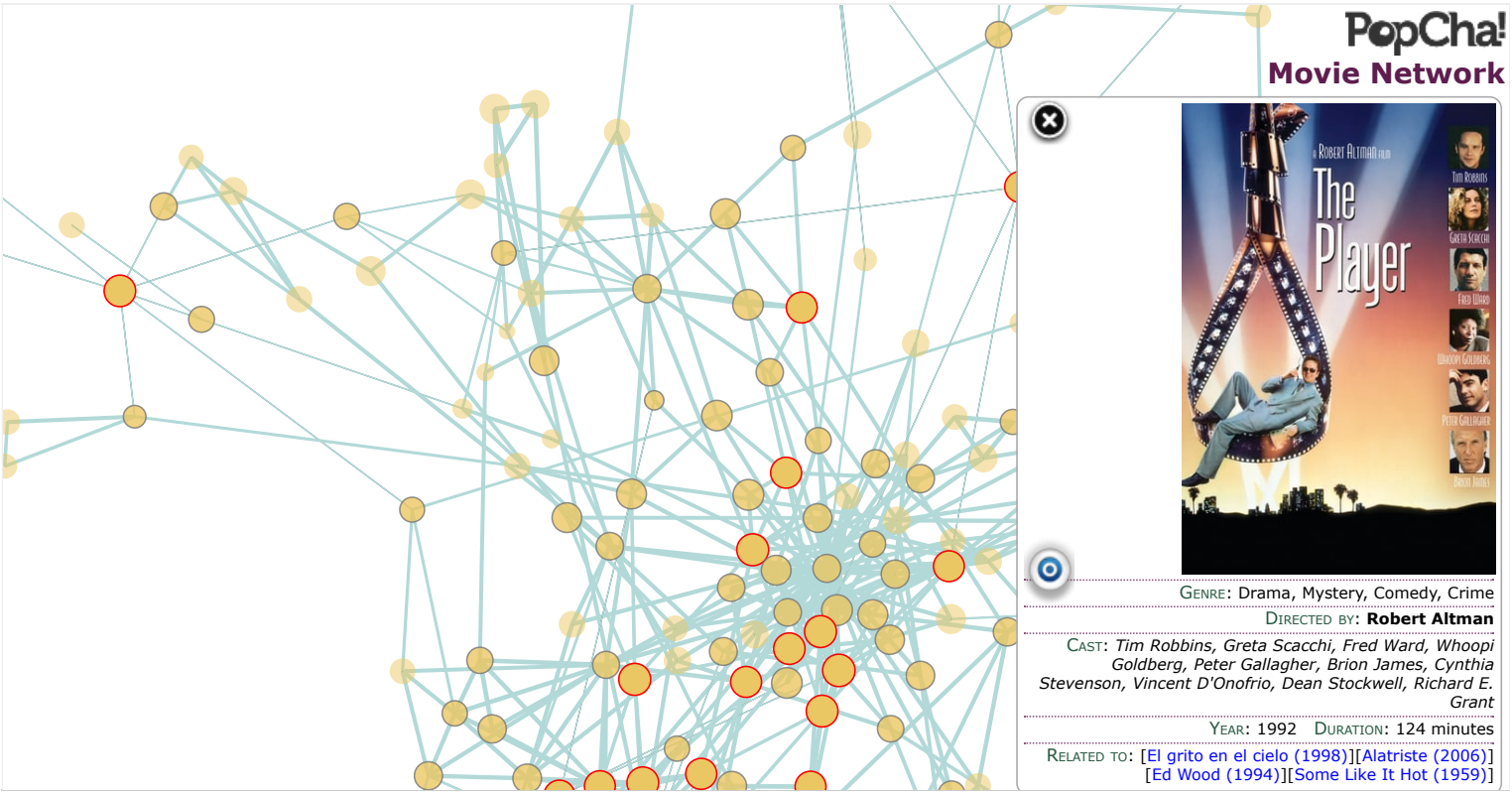
Paulo Villegas's Block 9686202
Updated November 16, 2018



# 9686202




# The PopCha! Movie Network

Open

A visualization of similarity between movies by means of a network graph

- Similarity data taken from PopCha! recommender engine (R*)
- Movie info from The Movie Database
- Visualization done in JavaScript with D3.js

## index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>PopCha! Movie Network</title>
    <link rel="stylesheet" href="movie-network.css"/>
    <link rel="shortcut icon" href="popcha.ico" type="image/x-icon">
    <script>
      // Sniff MSIE version
      // http://james.padolsey.com/javascript/detect-ie-in-js-using-conditional-comments/
      var ie = ( function() {
        var undef,
```

```
        v = 3,
        div = document.createElement('div'),
        all = div.getElementsByTagName('i');
        while (
         div.innerHTML='<!--[if gt IE ' + (++v) + ']><i></i><![endif]-->',all[0]
        );
        return v > 4 ? v : undef;
      }() );

      function takeAction() {
        if( ie && ie < 9 ) {
    D3notok();
        } else {
          // Load D3.js, and once loaded do our stuff
          var head = document.getElementsByTagName('head')[0];
          var script = document.createElement('script');
          script.type = 'text/javascript';
    script.src = "http://d3js.org/d3.v3.min.js";
          script.addEventListener('load', D3ok, false);
          script.onload = "D3ok();";
    head.appendChild(script);
       }
     }
    </script>
</head>
<body onload="takeAction();">

  <div id="nocontent">
    <h1>Sadly your browser is not compatible with this site</h1>

    <div>You can use <a href="http://www.google.com/chrome/">Google
    Chrome</a>, <a href="http://www.mozilla.org/firefox">Mozilla Firefox</a>
    or <a href="http://windows.microsoft.com/en-us/internet-explorer/download-ie">Microsoft
    Internet Explorer (v9 or above)</a> to access the PopCha Movie
    Network</div>

  </div>

  <div id="movieNetwork"></div>

  <div id="sidepanel">

    <div id="title">
      <img src="popcha-100.png"/><br/>Movie Network<br/>

      <!-- <a href="javascript:void(0);"
              onClick="zoomCall(0.5);" style="pointer-events: all;">+</a>
      <a href="javascript:void(0);"
         onClick="zoomCall(-0.5);" style="pointer-events: all;">-</a> -->

      <img id="helpIcon"
           src="help-question-48.png" title="Click for help"
           onClick="toggleDiv('help');"/>

    </div>

    <div id="help" class="panel_off">
      <img src="close.png" style="float: right;"
           onClick="toggleDiv('help');"/>
      This is a subset of the movie network produced from data in the
```

```
      mobile <a href="https://www.popcha.tv">PopCha!</a> app. Movies
      relate to each other <em>as they get picked by PopCha! users</em>
      (note that movie details are <strong>not</strong> used to build the
      graph).

      <ul>
        <li>Hover over a node to see the movie title, and to highlight
        the movies related to it.</li>

        <li>Click on a node to show movie details. It will open up a
        side panel.</li>

        <li>On the movie details panel, you can click on the <em>target</em>
        icon to center the graph on that movie. And clicking on the link to
        a related movie will move to that movie (in the graph and in the
        panel)</li>

        <li>You can use you usual browser controls to zoom and pan
        over the graph. If you zoom in when centred on a movie, the
        graph will expand around it.</li>

        <li>Above a certain zoom level, movie titles are automatically
        displayed for all the nodes.</li>
      </ul>
      For additional information, check the <strong><a href="" onClick="return toggleDiv('faq');">F
    </div>

    <div id="movieInfo" class="panel_off"></div>
  </div>

  <div id="faq" class="panel_off">
    <div id="close_faq">
      <img src="close.png" onClick="toggleDiv('faq');"/>
    </div>
    <dl>

      <dt>Where does this come from?</dt>

      <dd>The <a href="https://www.popcha.tv">PopCha!</a> mobile app
      lets users search for movies, view their details, rate them, add
      them to bags to build personalized movie lists and receive
      notifications when they are available in theaters, TV or Video
      on Demand platforms. Its recommendation engine allows proposing
      users movies they might be interested in. PopCha! users have so
      far currently rated, added to bags or otherwise showed interest
      for more than 11,000 movies</dd>

      <dt>And how was this concrete "movie network" made?</dt>
      <dd>This is how we did it:
        <ol style="margin: 0px;">
          <li> We selected the movies with the top ratings (the most
          highly valued movies for PopCha! users). Those were our
          initial seed, and are the nodes with a red border in the
          graph. As you can see, they are mostly the "usual suspects",
          but with also a few non-conventional movies added in.</li>

          <li>Then we used our recommendation engine to find the
          movies most similar to them. Not by looking at their genre,
          director or cast, but by checking how users interact with
          them. That is, two movies are considered similar just by
```

```
          checking if users rate them equally, or usually appear
          together in their bags. Those movies make up the second
          level in our network, marked with a grey border.</li>

          <li>Finally, we rounded up the graph by adding a third
          level, computed as the movies most similar to the ones in
          the second level</li>
        </ol>
      </dd>

      <dt>Ok, but <em>how</em> was the graph actually computed?</dt>

      <dd>For those of you technically oriented, the links between
      movies were extracted as item-item similarity scores computed in
      our Collaborative Filtering engine, using all events produced by
      users of the PopCha! Android/iOS apps. Those links were then fed
      to a <a
      href="https://github.com/mbostock/d3/wiki/Force-Layout">force-directed
      graph layout</a> in <strong>D3.js</strong>.

        <br>No movie metadata (title, director, cast, whatever) was used or harmed in any way durin
      <dt>What do the graph elements mean?</dt>

      <dd>Each circle is a movie; it is linked to the set of movies
      most similar to it according to our engine (and therefore to
      PopCha! users). The size of the circle represents how it is
      valued by PopCha! users; the set of 25 most valued movies is
      highlighted with a red border. <br/>

When a movie is selected (hovering or clicking) it is shown as purple
in the network. The set of movies linked to it (the ones more similar
to it) are also highlighted, this time in blue.<br/>

The width of the link between two movies reflect how similar they are
(in terms of user response to them); this similarity translates to the
force attracting the nodes (so that two very similar movies will tend
to stay close, within the constraints imposed by the rest of the
forces from the other nodes).<br/>

The force-directed layout is an iterative algorithm; each time the
page is loaded the rendering starts anew, and converges towards a
final stable state. There is some randomness in it, so that no two
final states are equal (but relative positioning of movies, and the
formed clusters, should be similar).


<dt>What is the resulting structure?</dt>
<dd>The graph is heavily connected: there are connections between many of the movies in the network


There is a strong central cluster containing some of the most important movies (it is so dense that

Outside the graph core we can also find other areas with semantic meaning (e.g. the set of <em><a h
    </dl>
    <dt>Credits</dt>
    <dd><em>
      Movie statistics by the <a href="https://www.popcha.tv">PopCha! team</a><br/>
      Graph computation & visualization by Paulo Villegas<br/>

      Movie details from <a href="http://www.themoviedb.org/">The Movie Database</a>
```

```
</em></dd>
  </div>

  <script src="movie-network.js"></script>
</body>
</html>
```

## movie-network.css

```
/* ------------------------------------------------------------------------
   (c) Telefónica I+D, 2013
   Author: Paulo Villegas
   ------------------------------------------------------------------------ */


body {
    background-color: #f6f6f6;
}

/* HREF links */

a {
    color: blue;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

a:visited {
    color: blue;
}

/* ..................................................................... */
/* The top-right side panel and its title child */

div#sidepanel {
    position: absolute;
    pointer-events: none;
    top: 0px;
    right: 0px;
}

div#title {
    margin: 4px 2px 6px 0px;
    border-width: 0px;
    padding: 0px;
    text-align: right;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 18px;
    line-height: 18px;
    color: #5D1D4D;
    font-weight: bold;
    pointer-events: none;
}

div#title img {
```

```
    border-width: 0px;
    margin: 0px;
}


img#helpIcon {
    position: absolute;
    right: 6px;
    top: 100px;
    pointer-events: all;
    cursor: help;
}


/* ..................................................................... */
/* On/off toggles for the help/info panels */

div.panel_off {
    visibility: hidden;
    pointer-events: none;
}

div.panel_on {
    visibility: visible;
    pointer-events: all;
}


/* ..................................................................... */
/* Help boxes */

div#help {
    margin: 6px;
    padding: 4px;
    background-color: #DEDFA3;
    position: absolute;
    width: 420px;
    right: 0px;
    z-index: 1001;
    border: 1px solid #989970;

    font-family: Georgia, Times New Roman, Times, serif;
    font-size: 11pt;

    -moz-transition: visibility 1.2s;
    -o-transition: visibility 1.2s;
    -webkit-transition: visibility 1.2s;
    transition: visibility 1.2s;

    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    border-radius: 10px;

    -webkit-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
    -moz-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
    box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
}

div#help ul {
    margin: 0.5em 0em 0.5em 0em;
```

```
    padding-left: 1.5em;

}

div#help li {
    margin: 0em;
    padding: 0px;
}

div#faq {
    position: absolute;
    top: 6px;
    left: 6px;
    padding: 4px;
    width: 860px;
    height: 600px;
    overflow-y: auto;
    z-index: 1002;
    background: #E5E4D6;

    font-size: 11pt;

    border: solid 1px #aaa;
    border-radius: 8px;
    -webkit-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
    -moz-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
    box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
}

div#faq dt {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 13pt;
    color: #5D1D4D;
    font-weight: bold;
    margin-top: .4em;
}

div#faq dd {
    font-family: Georgia, Times New Roman, Times, serif;
    padding-left: 0px;
    margin-left: 0.7em;
}

div#close_faq {
    position: fixed;
    margin-left: 820px;
    margin-top: 0px;
    padding: 0px;
}


/* ..................................................................... */

div#nocontent {
    visibility: hidden;
    pointer-events: none;
    position: absolute;
    width: 600px;
    height: 200px;
    top: 200px;
```

```
    left: 200px;
    background: #C3B091;
    border: solid 2px #a00;
    border-radius: 8px;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 20px;
    text-align: center;
    vertical-align: center;
    padding: 12px;
}

div#nocontent h1 {
    margin: 1em;
    color: red;
    font-size: 24px;
    font-weight: bold;
}

/* ..................................................................... */
/* Movie details panel */

div#movieInfo {
    position: relative;
    right: 4px;
    cursor: text;
    width: 300px;
    z-index: 1000;
    background: #E5E4D6;
    border: solid 1px #aaa;
    border-radius: 8px;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 10px;
    padding: 4px;
    text-align: right;
}

div#movieInfo div#cover {
    text-align: left;
    height: 300px;
}

div#movieInfo div.t {
    font-size: 14px;
    font-weight: bold;
}

div#movieInfo img.cover {
    margin-bottom: 6px;
    position: absolute;
    right: 3px;
}

div#movieInfo img.action {
    cursor: pointer;
    position: absolute;
}

div#movieInfo div.f {
    border-top: 1px dotted #8E5981;
    margin-bottom: 3px;
```

```
    margin-top: 3px;
}

div#movieInfo span.d {
    font-weight: bold;
}

div#movieInfo span.c {
    font-style: italic;
}

div#movieInfo span.l {
    font-size: 11px;
    color: #24553E;
    font-variant: small-caps;
}

/* ..................................................................... */
/* SVG elements */

div#movieNetwork svg {
    background-color: white;
    cursor: move;
}

line.link {
    stroke: #B2D9D8;
}

circle {
    cursor: crosshair;
    fill: #EBC763;
}

circle.level1 {
    stroke: #f00;
}

circle.level2 {
    fill-opacity: 0.8;
    stroke-opacity: 0.8;
    stroke: #777;
}

circle.level3 {
    fill-opacity: 0.5;
    stroke-opacity: 0.5;
}

circle.sibling {
    /*fill: blue;*/
    fill: #455EE8;
}

circle.main {
    /*fill: red;*/
    fill: #732A9A;
    fill-opacity: 1.0;
}
```

```
/* ..................................................................... */
/* Graph labels */

g.gLabel {
    font: 10px sans-serif;
    font-weight: normal;
    visibility: hidden;
}

g.on {
    visibility: visible;
}

g.zoomed {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 10px;
    font-weight: normal;
    text-align: center;
    color: #000;
    border: none;
    z-index: 0;
}

text {
    font: 10px sans-serif;
    font-weight: normal;
    stroke-opacity: 1.0;
}

text.nlabel {
    /*stroke: #000000;*/
}

text.nshadow {
    stroke: #fff;
    stroke-width: 3px;
    /*stroke-opacity: 0.5;*/
    /*visibility: hidden;*/
}

text.main {
    font: 12px sans-serif;
    font-weight: bold;
}


/* ..................................................................... */

/* no longer used */

.nlabel_on {
    visibility: visible;
    font-size: 12px;
    opacity: 1.0;
    fill: #101000;
    stroke: #ffffff;
    font-weight: bold;
}

.tooltip{
```

```
    position: absolute;
    width: 200px;
    height: 50px;
    padding: 8px;
    font: 15px Helvetica Neue;
    background: #FFF;
    border: solid 1px #aaa;
    border-radius: 8px;
    pointer-events: none;
    z-index:1000;
    text-align:center;
    background: rgba(222,223,163,0.8);
}
```

## movie-network.js

```
/* ------------------------------------------------------------------------
   (c) Telefónica I+D, 2013
   Author: Paulo Villegas

   This script is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>.
   ------------------------------------------------------------------------ */


// For MSIE < 9, forget it
function D3notok() {
  document.getElementById('sidepanel').style.visibility = 'hidden';
  var nocontent = document.getElementById('nocontent');
  nocontent.style.visibility = 'visible';
  nocontent.style.pointerEvents = 'all';
  var t = document.getElementsByTagName('body');
  var body = document.getElementsByTagName('body')[0];
  body.style.backgroundImage = "url('movie-network-screenshot-d.png')";
  body.style.backgroundRepeat = "no-repeat";
}

// ---------------------------------------------------------------------
// A number of forward declarations. These variables need to be defined since
// they are attached to static code in HTML. But we cannot define them yet
// since they need D3.js stuff. So we put placeholders.


// Highlight a movie in the graph. It is a closure within the d3.json() call.
var selectMovie = undefined;
```

```
// Change status of a panel from visible to hidden or viceversa
var toggleDiv = undefined;

// Clear all help boxes and select a movie in network and in movie details panel
var clearAndSelect = undefined;


// The call to set a zoom value -- currently unused
// (zoom is set via standard mouse-based zooming)
var zoomCall = undefined;


// ---------------------------------------------------------------------

// Do the stuff -- to be called after D3.js has loaded
function D3ok() {

  // Some constants
  var WIDTH = 960,
      HEIGHT = 600,
      SHOW_THRESHOLD = 2.5;

  // Variables keeping graph state
  var activeMovie = undefined;
  var currentOffset = { x : 0, y : 0 };
  var currentZoom = 1.0;

  // The D3.js scales
  var xScale = d3.scale.linear()
    .domain([0, WIDTH])
    .range([0, WIDTH]);
  var yScale = d3.scale.linear()
    .domain([0, HEIGHT])
    .range([0, HEIGHT]);
  var zoomScale = d3.scale.linear()
    .domain([1,6])
    .range([1,6])
    .clamp(true);

/* ..................................................................... */

  // The D3.js force-directed layout
  var force = d3.layout.force()
    .charge(-320)
    .size( [WIDTH, HEIGHT] )
    .linkStrength( function(d,idx) { return d.weight; } );

  // Add to the page the SVG element that will contain the movie network
  var svg = d3.select("#movieNetwork").append("svg:svg")
    .attr('xmlns','http://www.w3.org/2000/svg')
    .attr("width", WIDTH)
    .attr("height", HEIGHT)
    .attr("id","graph")
    .attr("viewBox", "0 0 " + WIDTH + " " + HEIGHT )
    .attr("preserveAspectRatio", "xMidYMid meet");

  // Movie panel: the div into which the movie details info will be written
  movieInfoDiv = d3.select("#movieInfo");

  /* ..................................................................... */
```

```
// Get the current size & offset of the browser's viewport window
function getViewportSize( w ) {
  var w = w || window;
  if( w.innerWidth != null )
    return { w: w.innerWidth,
       h: w.innerHeight,
       x : w.pageXOffset,
       y : w.pageYOffset };
  var d = w.document;
  if( document.compatMode == "CSS1Compat" )
    return { w: d.documentElement.clientWidth,
       h: d.documentElement.clientHeight,
       x: d.documentElement.scrollLeft,
       y: d.documentElement.scrollTop };
  else
    return { w: d.body.clientWidth,
       h: d.body.clientHeight,
       x: d.body.scrollLeft,
       y: d.body.scrollTop};
}



function getQStringParameterByName(name) {
  var match = RegExp('[?&]' + name + '=([^&]*)').exec(window.location.search);
  return match && decodeURIComponent(match[1].replace(/\+/g, ' '));
}


/* Change status of a panel from visible to hidden or viceversa
   id: identifier of the div to change
   status: 'on' or 'off'. If not specified, the panel will toggle status
*/
toggleDiv = function( id, status ) {
  d = d3.select('div#'+id);
  if( status === undefined )
    status = d.attr('class') == 'panel_on' ? 'off' : 'on';
  d.attr( 'class', 'panel_' + status );
  return false;
}


/* Clear all help boxes and select a movie in the network and in the
   movie details panel
*/
clearAndSelect = function (id) {
  toggleDiv('faq','off');
  toggleDiv('help','off');
  selectMovie(id,true); // we use here the selectMovie() closure
}


/* Compose the content for the panel with movie details.
   Parameters: the node data, and the array containing all nodes
*/
function getMovieInfo( n, nodeArray ) {
  info = '<div id="cover">';
  if( n.cover )
    info += '<img class="cover" height="300" src="' + n.cover + '" title="' + n.label + '"/>';
```

```
  else
    info += '<div class=t style="float: right">' + n.title + '</div>';
  info +=
  '<img src="close.png" class="action" style="top: 0px;" title="close panel" onClick="toggleDiv(\
  '<img src="target-32.png" class="action" style="top: 280px;" title="center graph on movie" oncl

  info += '<br/></div><div style="clear: both;">'
  if( n.genre )
    info += '<div class=f><span class=l>Genre</span>: <span class=g>'
         + n.genre + '</span></div>';
  if( n.director )
    info += '<div class=f><span class=l>Directed by</span>: <span class=d>'
         + n.director + '</span></div>';
  if( n.cast )
    info += '<div class=f><span class=l>Cast</span>: <span class=c>'
         + n.cast + '</span></div>';
  if( n.duration )
    info += '<div class=f><span class=l>Year</span>: ' + n.year
         + '<span class=l style="margin-left:1em;">Duration</span>: '
         + n.duration + '</div>';
  if( n.links ) {
    info += '<div class=f><span class=l>Related to</span>: ';
    n.links.forEach( function(idx) {
info += '[<a href="javascript:void(0);" onclick="selectMovie('
     + idx + ',true);">' + nodeArray[idx].label + '</a>]'
    });
    info += '</div>';
  }
  return info;
}


// ****************************************************************************

d3.json(
  'movie-network-25-7-3.json',
  function(data) {

  // Declare the variables pointing to the node & link arrays
  var nodeArray = data.nodes;
  var linkArray = data.links;

  minLinkWeight =
    Math.min.apply( null, linkArray.map( function(n) {return n.weight;} ) );
  maxLinkWeight =
    Math.max.apply( null, linkArray.map( function(n) {return n.weight;} ) );

  // Add the node & link arrays to the layout, and start it
  force
    .nodes(nodeArray)
    .links(linkArray)
    .start();

  // A couple of scales for node radius & edge width
  var node_size = d3.scale.linear()
    .domain([5,10]) // we know score is in this domain
    .range([1,16])
    .clamp(true);
  var edge_width = d3.scale.pow().exponent(8)
    .domain( [minLinkWeight,maxLinkWeight] )
```

```
    .range([1,3])
    .clamp(true);

/* Add drag & zoom behaviours */
svg.call( d3.behavior.drag()
    .on("drag",dragmove) );
svg.call( d3.behavior.zoom()
    .x(xScale)
    .y(yScale)
    .scaleExtent([1, 6])
    .on("zoom", doZoom) );

// ------- Create the elements of the layout (links and nodes) ------

var networkGraph = svg.append('svg:g').attr('class','grpParent');

// links: simple lines
var graphLinks = networkGraph.append('svg:g').attr('class','grp gLinks')
  .selectAll("line")
  .data(linkArray, function(d) {return d.source.id+'-'+d.target.id;} )
  .enter().append("line")
  .style('stroke-width', function(d) { return edge_width(d.weight);} )
  .attr("class", "link");

// nodes: an SVG circle
var graphNodes = networkGraph.append('svg:g').attr('class','grp gNodes')
  .selectAll("circle")
  .data( nodeArray, function(d){ return d.id; } )
  .enter().append("svg:circle")
  .attr('id', function(d) { return "c" + d.index; } )
  .attr('class', function(d) { return 'node level'+d.level;} )
  .attr('r', function(d) { return node_size(d.score || 3); } )
  .attr('pointer-events', 'all')
  //.on("click", function(d) { highlightGraphNode(d,true,this); } )
  .on("click", function(d) { showMoviePanel(d); } )
  .on("mouseover", function(d) { highlightGraphNode(d,true,this);  } )
  .on("mouseout",  function(d) { highlightGraphNode(d,false,this); } );

// labels: a group with two SVG text: a title and a shadow (as background)
var graphLabels = networkGraph.append('svg:g').attr('class','grp gLabel')
  .selectAll("g.label")
  .data( nodeArray, function(d){return d.label} )
  .enter().append("svg:g")
  .attr('id', function(d) { return "l" + d.index; } )
  .attr('class','label');

shadows = graphLabels.append('svg:text')
  .attr('x','-2em')
  .attr('y','-.3em')
  .attr('pointer-events', 'none') // they go to the circle beneath
  .attr('id', function(d) { return "lb" + d.index; } )
  .attr('class','nshadow')
  .text( function(d) { return d.label; } );

labels = graphLabels.append('svg:text')
  .attr('x','-2em')
  .attr('y','-.3em')
  .attr('pointer-events', 'none') // they go to the circle beneath
  .attr('id', function(d) { return "lf" + d.index; } )
  .attr('class','nlabel')
```

```
      .text( function(d) { return d.label; } );


    /* ---------------------------------------------------------------------- */
    /* Select/unselect a node in the network graph.
       Parameters are:
       - node: data for the node to be changed,
       - on: true/false to show/hide the node
    */
    function highlightGraphNode( node, on )
    {
      //if( d3.event.shiftKey ) on = false; // for debugging

      // If we are to activate a movie, and there's already one active,
      // first switch that one off
      if( on && activeMovie !== undefined ) {
  highlightGraphNode( nodeArray[activeMovie], false );
      }

      // locate the SVG nodes: circle & label group
      circle = d3.select( '#c' + node.index );
      label  = d3.select( '#l' + node.index );

      // activate/deactivate the node itself
      circle
  .classed( 'main', on );
      label
  .classed( 'on', on || currentZoom >= SHOW_THRESHOLD );
      label.selectAll('text')
  .classed( 'main', on );

      // activate all siblings
      Object(node.links).forEach( function(id) {
  d3.select("#c"+id).classed( 'sibling', on );
  label = d3.select('#l'+id);
  label.classed( 'on', on || currentZoom >= SHOW_THRESHOLD );
  label.selectAll('text.nlabel')
    .classed( 'sibling', on );
      } );

      // set the value for the current active movie
      activeMovie = on ? node.index : undefined;
    }


    /* ---------------------------------------------------------------------- */
    /* Show the details panel for a movie AND highlight its node in
       the graph. Also called from outside the d3.json context.
       Parameters:
       - new_idx: index of the movie to show
       - doMoveTo: boolean to indicate if the graph should be centered
         on the movie
    */
    selectMovie = function( new_idx, doMoveTo ) {

      // do we want to center the graph on the node?
      doMoveTo = doMoveTo || false;
      if( doMoveTo ) {
  s = getViewportSize();
  width  = s.w<WIDTH ? s.w : WIDTH;
```

```
height = s.h<HEIGHT ? s.h : HEIGHT;
offset = { x : s.x + width/2  - nodeArray[new_idx].x*currentZoom,
     y : s.y + height/2 - nodeArray[new_idx].y*currentZoom };
repositionGraph( offset, undefined, 'move' );
    }
    // Now highlight the graph node and show its movie panel
    highlightGraphNode( nodeArray[new_idx], true );
    showMoviePanel( nodeArray[new_idx] );
  }


  /* ------------------------------------------------------------------------ */
  /* Show the movie details panel for a given node
   */
  function showMoviePanel( node ) {
    // Fill it and display the panel
    movieInfoDiv
.html( getMovieInfo(node,nodeArray) )
.attr("class","panel_on");
  }


  /* ------------------------------------------------------------------------ */
  /* Move all graph elements to its new positions. Triggered:
     - on node repositioning (as result of a force-directed iteration)
     - on translations (user is panning)
     - on zoom changes (user is zooming)
     - on explicit node highlight (user clicks in a movie panel link)
     Set also the values keeping track of current offset & zoom values
  */
  function repositionGraph( off, z, mode ) {

    // do we want to do a transition?
    var doTr = (mode == 'move');

    // drag: translate to new offset
    if( off !== undefined &&
  (off.x != currentOffset.x || off.y != currentOffset.y ) ) {
g = d3.select('g.grpParent')
if( doTr )
  g = g.transition().duration(500);
g.attr("transform", function(d) { return "translate("+
          off.x+","+off.y+")" } );
currentOffset.x = off.x;
currentOffset.y = off.y;
    }

    // zoom: get new value of zoom
    if( z === undefined ) {
if( mode != 'tick' )
  return; // no zoom, no tick, we don't need to go further
z = currentZoom;
    }
    else
currentZoom = z;

    // move edges
    e = doTr ? graphLinks.transition().duration(500) : graphLinks;
    e
.attr("x1", function(d) { return z*(d.source.x); })
```

```
        .attr("y1", function(d) { return z*(d.source.y); })
        .attr("x2", function(d) { return z*(d.target.x); })
        .attr("y2", function(d) { return z*(d.target.y); });

    // move nodes
    n = doTr ? graphNodes.transition().duration(500) : graphNodes;
    n
.attr("transform", function(d) { return "translate("
          +z*d.x+","+z*d.y+")" } );
    // move labels
    l = doTr ? graphLabels.transition().duration(500) : graphLabels;
    l
.attr("transform", function(d) { return "translate("
          +z*d.x+","+z*d.y+")" } );
  }


  /* ------------------------------------------------------------------------ */
  /* Perform drag
   */
  function dragmove(d) {
    offset = { x : currentOffset.x + d3.event.dx,
   y : currentOffset.y + d3.event.dy };
    repositionGraph( offset, undefined, 'drag' );
  }


  /* ------------------------------------------------------------------------ */
  /* Perform zoom. We do "semantic zoom", not geometric zoom
   * (i.e. nodes do not change size, but get spread out or stretched
   * together as zoom changes)
   */
  function doZoom( increment ) {
    newZoom = increment === undefined ? d3.event.scale
        : zoomScale(currentZoom+increment);
    if( currentZoom == newZoom )
return; // no zoom change

    // See if we cross the 'show' threshold in either direction
    if( currentZoom<SHOW_THRESHOLD && newZoom>=SHOW_THRESHOLD )
svg.selectAll("g.label").classed('on',true);
    else if( currentZoom>=SHOW_THRESHOLD && newZoom<SHOW_THRESHOLD )
svg.selectAll("g.label").classed('on',false);

    // See what is the current graph window size
    s = getViewportSize();
    width  = s.w<WIDTH  ? s.w : WIDTH;
    height = s.h<HEIGHT ? s.h : HEIGHT;

    // Compute the new offset, so that the graph center does not move
    zoomRatio = newZoom/currentZoom;
    newOffset = { x : currentOffset.x*zoomRatio + width/2*(1-zoomRatio),
      y : currentOffset.y*zoomRatio + height/2*(1-zoomRatio) };

    // Reposition the graph
    repositionGraph( newOffset, newZoom, "zoom" );
  }

  zoomCall = doZoom;  // unused, so far
```

```
    /* ---------------------------------------------------------------------- */

    /* process events from the force-directed graph */
    force.on("tick", function() {
      repositionGraph(undefined,undefined,'tick');
    });

    /* A small hack to start the graph with a movie pre-selected */
    mid = getQStringParameterByName('id')
    if( mid != null )
      clearAndSelect( mid );
  });

} // end of D3ok()
```

## LICENSE