Collusion Detection Dashboard for Healthcare Using Louvain Algorithm

Here's a comprehensive approach to creating a collusion detection dashboard that visualizes connections between hospitals, doctors, and member associations using the Louvain community detection algorithm.

Key Visualizations for Your Dashboard

1. **Network Graph View**

* **Primary visualization**: Force-directed graph showing nodes (hospitals, doctors, members) and edges (connections)
* **Features**:
  + Node size: Proportional to billed amounts
  + Node color: Represents Louvain-detected communities
  + Edge thickness: Weighted by connection strength (number of claims, referral frequency)
  + Hover details: Show specific claim counts, dollar amounts, and referral patterns

2. **Community Overview Heatmap**

* Matrix showing connection strength between entities
* X-axis: Hospitals/Doctors
* Y-axis: Member associations
* Color intensity: Represents total billed amounts or claim frequency

3. **Temporal Analysis View**

* Line chart showing billing patterns over time
* Highlight synchronized spikes in claims among community members
* Animate network graph to show evolution of connections over time

4. **Geospatial View**

* Map overlay showing geographic distribution of colluding entities
* Cluster unusually high concentrations of connections in specific areas

5. **Referral Pattern Sunburst Chart**

* Hierarchical view of referral patterns
* Inner ring: Hospitals
* Middle ring: Doctors
* Outer ring: Member associations
* Arc width: Number of referrals

6. **Billing Anomaly Radar Chart**

* Compare billing patterns of detected communities against benchmarks
* Axes: Average claim amount, frequency, procedure codes, etc.

Specific Connection Metrics to Visualize

1. **Claim-Based Connections**:
   * Number of shared claims between entities
   * Unusually high claim frequencies between specific pairs
2. **Billing Amount Connections**:
   * Total billed amounts between entity pairs
   * Percentage of total billing going to specific groups
3. **Referral Patterns**:
   * Directed graphs showing referral flows
   * Circular referral patterns (A→B→C→A)
4. **Procedure Code Analysis**:
   * Unusual concentration of specific procedure codes within communities
   * Comparison to normal code distribution

Implementation Recommendations

1. **Data Preparation**:
   * Create weighted edges based on:
     + Claim frequency
     + Billed amounts
     + Referral counts
     + Temporal proximity of claims
2. **Louvain Optimization**:
   * Experiment with different resolution parameters
   * Consider multi-level community detection
   * Add node attributes (specialty, location) to enhance detection
3. **Interactive Features**:
   * Filter by time period, dollar thresholds, or geographic region
   * Drill-down capability from communities to individual entities
   * Side-by-side comparison of suspicious vs normal communities
4. **Alert System**:
   * Visual indicators for communities that exceed thresholds
   * Scoring system for collusion likelihood

Would you like me to elaborate on any particular aspect of this dashboard design or the implementation approach?

can you generate full html and javascript code for this dashboard. It should contain different section to provide analysis to end user.

Healthcare Collusion Detection Dashboard

Here's a complete HTML and JavaScript implementation for a collusion detection dashboard using the Louvain algorithm. This dashboard includes multiple visualization sections to analyze connections between hospitals, doctors, and member associations.

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Healthcare Collusion Detection Dashboard</title>

<script src="https://d3js.org/d3.v7.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script src="https://cdn.jsdelivr.net/npm/echarts@5.4.3/dist/echarts.min.js"></script>

<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

margin: 0;

padding: 0;

background-color: #f5f7fa;

color: #333;

}

.dashboard {

display: grid;

grid-template-columns: 1fr;

gap: 20px;

padding: 20px;

max-width: 1800px;

margin: 0 auto;

}

.header {

background-color: #2c3e50;

color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.card {

background-color: white;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

padding: 20px;

margin-bottom: 20px;

}

.card-title {

font-size: 1.2em;

font-weight: bold;

margin-bottom: 15px;

color: #2c3e50;

border-bottom: 1px solid #eee;

padding-bottom: 10px;

}

.grid-2 {

display: grid;

grid-template-columns: 1fr 1fr;

gap: 20px;

}

.grid-3 {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

gap: 20px;

}

.chart-container {

width: 100%;

height: 400px;

}

#network-graph {

height: 600px;

}

#map {

height: 500px;

border-radius: 8px;

}

.controls {

display: flex;

justify-content: space-between;

margin-bottom: 20px;

flex-wrap: wrap;

gap: 10px;

}

.control-group {

display: flex;

gap: 10px;

align-items: center;

}

select, input, button {

padding: 8px 12px;

border-radius: 4px;

border: 1px solid #ddd;

background-color: white;

}

button {

background-color: #3498db;

color: white;

border: none;

cursor: pointer;

transition: background-color 0.3s;

}

button:hover {

background-color: #2980b9;

}

.community-info {

margin-top: 20px;

padding: 15px;

background-color: #f8f9fa;

border-radius: 8px;

border-left: 4px solid #3498db;

}

.risk-indicator {

display: inline-block;

padding: 3px 8px;

border-radius: 4px;

font-size: 0.8em;

font-weight: bold;

}

.high-risk {

background-color: #e74c3c;

color: white;

}

.medium-risk {

background-color: #f39c12;

color: white;

}

.low-risk {

background-color: #2ecc71;

color: white;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 15px;

}

th, td {

padding: 10px;

text-align: left;

border-bottom: 1px solid #ddd;

}

th {

background-color: #f2f2f2;

}

tr:hover {

background-color: #f5f5f5;

}

.tooltip {

position: absolute;

padding: 10px;

background: rgba(0, 0, 0, 0.8);

color: white;

border-radius: 5px;

pointer-events: none;

font-size: 12px;

z-index: 1000;

}

</style>

</head>

<body>

<div class="dashboard">

<div class="header">

<h1>Healthcare Collusion Detection Dashboard</h1>

<p>Analyzing connections between hospitals, doctors, and member associations using Louvain community detection</p>

</div>

<div class="controls">

<div class="control-group">

<label for="time-period">Time Period:</label>

<select id="time-period">

<option value="all">All Time</option>

<option value="1y">Last Year</option>

<option value="6m">Last 6 Months</option>

<option value="3m">Last 3 Months</option>

<option value="1m">Last Month</option>

</select>

</div>

<div class="control-group">

<label for="entity-type">Focus Entity:</label>

<select id="entity-type">

<option value="all">All Entities</option>

<option value="hospital">Hospitals</option>

<option value="doctor">Doctors</option>

<option value="member">Member Associations</option>

</select>

</div>

<div class="control-group">

<label for="min-amount">Min Amount ($):</label>

<input type="number" id="min-amount" value="10000" min="0">

</div>

<div class="control-group">

<button id="apply-filters">Apply Filters</button>

<button id="reset-filters">Reset</button>

</div>

</div>

<div class="card">

<div class="card-title">Network Graph of Detected Communities</div>

<div id="network-graph" class="chart-container"></div>

<div class="community-info">

<h3>Selected Community Analysis</h3>

<p id="community-details">Click on a node or community to see details</p>

<div id="community-metrics" style="display: none;">

<div class="grid-3">

<div>

<h4>Risk Assessment</h4>

<p>Collusion Probability: <span id="collusion-prob" class="risk-indicator high-risk">High</span></p>

<p>Community Size: <span id="community-size">5</span> entities</p>

<p>Total Billed Amount: $<span id="total-billed">1,234,567</span></p>

</div>

<div>

<h4>Anomaly Indicators</h4>

<p>Claim Frequency: <span id="claim-freq">3.5x</span> expected</p>

<p>Procedure Concentration: <span id="proc-concentration">82%</span> similar codes</p>

<p>Referral Circularity: <span id="referral-circularity">High</span></p>

</div>

<div>

<h4>Geographic Spread</h4>

<p>Locations: <span id="geo-spread">3</span> distinct regions</p>

<p>Average Distance: <span id="avg-distance">12.5</span> miles</p>

<p>Clustering Factor: <span id="clustering-factor">0.87</span></p>

</div>

</div>

</div>

</div>

</div>

<div class="grid-2">

<div class="card">

<div class="card-title">Community Billing Patterns</div>

<div id="billing-chart" class="chart-container"></div>

</div>

<div class="card">

<div class="card-title">Referral Patterns Sunburst</div>

<div id="referral-chart" class="chart-container"></div>

</div>

</div>

<div class="card">

<div class="card-title">Geographic Distribution of Suspect Communities</div>

<div id="map"></div>

</div>

<div class="grid-3">

<div class="card">

<div class="card-title">Top Risky Communities</div>

<div id="risk-chart" class="chart-container"></div>

</div>

<div class="card">

<div class="card-title">Procedure Code Analysis</div>

<div id="procedure-chart" class="chart-container"></div>

</div>

<div class="card">

<div class="card-title">Temporal Claim Patterns</div>

<div id="temporal-chart" class="chart-container"></div>

</div>

</div>

<div class="card">

<div class="card-title">Detailed Community Members</div>

<table id="community-table">

<thead>

<tr>

<th>Entity Type</th>

<th>Name/ID</th>

<th>Claims</th>

<th>Billed Amount</th>

<th>Referrals</th>

<th>Risk Score</th>

</tr>

</thead>

<tbody>

*<!-- Will be populated by JavaScript -->*

</tbody>

</table>

</div>

</div>

<div class="tooltip" id="graph-tooltip"></div>

<script>

// Sample data - in a real application, this would come from an API

const sampleData = {

nodes: [

// Hospitals

{ id: "h1", name: "City General Hospital", type: "hospital", size: 45, community: 0, lat: 40.7128, lng: -74.0060, claims: 1200, amount: 4500000, risk: 0.85 },

{ id: "h2", name: "Regional Medical Center", type: "hospital", size: 38, community: 1, lat: 34.0522, lng: -118.2437, claims: 950, amount: 3800000, risk: 0.72 },

{ id: "h3", name: "Community Hospital", type: "hospital", size: 28, community: 2, lat: 41.8781, lng: -87.6298, claims: 700, amount: 2800000, risk: 0.45 },

{ id: "h4", name: "University Hospital", type: "hospital", size: 32, community: 3, lat: 39.9526, lng: -75.1652, claims: 800, amount: 3200000, risk: 0.35 },

// Doctors

{ id: "d1", name: "Dr. Smith (Cardiology)", type: "doctor", size: 25, community: 0, lat: 40.7128, lng: -74.0060, claims: 300, amount: 1200000, risk: 0.82 },

{ id: "d2", name: "Dr. Johnson (Ortho)", type: "doctor", size: 22, community: 0, lat: 40.7128, lng: -74.0060, claims: 280, amount: 1100000, risk: 0.78 },

{ id: "d3", name: "Dr. Williams (Neuro)", type: "doctor", size: 20, community: 1, lat: 34.0522, lng: -118.2437, claims: 250, amount: 1000000, risk: 0.65 },

{ id: "d4", name: "Dr. Brown (Oncology)", type: "doctor", size: 18, community: 1, lat: 34.0522, lng: -118.2437, claims: 220, amount: 900000, risk: 0.60 },

{ id: "d5", name: "Dr. Davis (Family)", type: "doctor", size: 15, community: 2, lat: 41.8781, lng: -87.6298, claims: 180, amount: 700000, risk: 0.40 },

{ id: "d6", name: "Dr. Miller (Pediatrics)", type: "doctor", size: 12, community: 3, lat: 39.9526, lng: -75.1652, claims: 150, amount: 600000, risk: 0.30 },

// Member Associations

{ id: "m1", name: "HealthPlus Members", type: "member", size: 30, community: 0, lat: 40.7128, lng: -74.0060, claims: 400, amount: 1500000, risk: 0.80 },

{ id: "m2", name: "CareFirst Associates", type: "member", size: 28, community: 0, lat: 40.7128, lng: -74.0060, claims: 380, amount: 1400000, risk: 0.75 },

{ id: "m3", name: "Wellness Partners", type: "member", size: 25, community: 1, lat: 34.0522, lng: -118.2437, claims: 350, amount: 1300000, risk: 0.68 },

{ id: "m4", name: "Prime Health Group", type: "member", size: 20, community: 2, lat: 41.8781, lng: -87.6298, claims: 300, amount: 1200000, risk: 0.42 },

{ id: "m5", name: "MediCare Alliance", type: "member", size: 18, community: 3, lat: 39.9526, lng: -75.1652, claims: 280, amount: 1100000, risk: 0.32 }

],

links: [

// Community 0 connections (high risk)

{ source: "h1", target: "d1", value: 25, type: "referral", amount: 500000 },

{ source: "h1", target: "d2", value: 22, type: "referral", amount: 450000 },

{ source: "d1", target: "m1", value: 30, type: "claim", amount: 600000 },

{ source: "d2", target: "m2", value: 28, type: "claim", amount: 550000 },

{ source: "m1", target: "h1", value: 20, type: "claim", amount: 400000 },

{ source: "m2", target: "h1", value: 18, type: "claim", amount: 350000 },

{ source: "d1", target: "m2", value: 15, type: "claim", amount: 300000 },

// Community 1 connections (medium risk)

{ source: "h2", target: "d3", value: 20, type: "referral", amount: 400000 },

{ source: "h2", target: "d4", value: 18, type: "referral", amount: 350000 },

{ source: "d3", target: "m3", value: 25, type: "claim", amount: 500000 },

{ source: "d4", target: "m3", value: 22, type: "claim", amount: 450000 },

// Community 2 connections (low risk)

{ source: "h3", target: "d5", value: 15, type: "referral", amount: 300000 },

{ source: "d5", target: "m4", value: 18, type: "claim", amount: 350000 },

// Community 3 connections (low risk)

{ source: "h4", target: "d6", value: 12, type: "referral", amount: 250000 },

{ source: "d6", target: "m5", value: 15, type: "claim", amount: 300000 },

// Some cross-community connections

{ source: "d3", target: "m1", value: 5, type: "claim", amount: 100000 },

{ source: "d5", target: "m2", value: 3, type: "claim", amount: 60000 }

],

communities: [

{ id: 0, name: "Community 1", risk: "high", members: ["h1", "d1", "d2", "m1", "m2"],

billingPattern: [1200000, 1350000, 1500000, 1450000, 1600000, 1750000, 1800000, 1850000, 1900000, 1950000, 2000000, 2100000],

procedureCodes: [

{code: "99214", count: 320, avgAmount: 120},

{code: "99213", count: 280, avgAmount: 90},

{code: "99232", count: 240, avgAmount: 80},

{code: "99233", count: 180, avgAmount: 110},

{code: "99285", count: 150, avgAmount: 350}

],

referralPattern: {

"h1": { "d1": 25, "d2": 22 },

"d1": { "m1": 30, "m2": 15 },

"d2": { "m2": 28 },

"m1": { "h1": 20 },

"m2": { "h1": 18 }

}

},

{ id: 1, name: "Community 2", risk: "medium", members: ["h2", "d3", "d4", "m3"],

billingPattern: [900000, 950000, 1000000, 1050000, 1100000, 1150000, 1200000, 1250000, 1300000, 1350000, 1400000, 1450000],

procedureCodes: [

{code: "99214", count: 280, avgAmount: 120},

{code: "99213", count: 240, avgAmount: 90},

{code: "99232", count: 200, avgAmount: 80},

{code: "99233", count: 160, avgAmount: 110},

{code: "99285", count: 120, avgAmount: 350}

],

referralPattern: {

"h2": { "d3": 20, "d4": 18 },

"d3": { "m3": 25 },

"d4": { "m3": 22 }

}

},

{ id: 2, name: "Community 3", risk: "low", members: ["h3", "d5", "m4"],

billingPattern: [600000, 650000, 700000, 750000, 800000, 850000, 900000, 950000, 1000000, 1050000, 1100000, 1150000],

procedureCodes: [

{code: "99214", count: 180, avgAmount: 120},

{code: "99213", count: 160, avgAmount: 90},

{code: "99232", count: 140, avgAmount: 80},

{code: "99233", count: 120, avgAmount: 110},

{code: "99285", count: 100, avgAmount: 350}

],

referralPattern: {

"h3": { "d5": 15 },

"d5": { "m4": 18 }

}

},

{ id: 3, name: "Community 4", risk: "low", members: ["h4", "d6", "m5"],

billingPattern: [500000, 550000, 600000, 650000, 700000, 750000, 800000, 850000, 900000, 950000, 1000000, 1050000],

procedureCodes: [

{code: "99214", count: 150, avgAmount: 120},

{code: "99213", count: 140, avgAmount: 90},

{code: "99232", count: 120, avgAmount: 80},

{code: "99233", count: 100, avgAmount: 110},

{code: "99285", count: 80, avgAmount: 350}

],

referralPattern: {

"h4": { "d6": 12 },

"d6": { "m5": 15 }

}

}

],

temporalData: {

months: ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"],

communityClaims: [

{name: "Community 1", data: [120, 135, 150, 145, 160, 175, 180, 185, 190, 195, 200, 210]},

{name: "Community 2", data: [90, 95, 100, 105, 110, 115, 120, 125, 130, 135, 140, 145]},

{name: "Community 3", data: [60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115]},

{name: "Community 4", data: [50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]}

]

}

};

// Initialize the dashboard

document.addEventListener('DOMContentLoaded', function() {

initNetworkGraph();

initBillingChart();

initReferralChart();

initMap();

initRiskChart();

initProcedureChart();

initTemporalChart();

initCommunityTable();

// Set up event listeners

document.getElementById('apply-filters').addEventListener('click', applyFilters);

document.getElementById('reset-filters').addEventListener('click', resetFilters);

});

function applyFilters() {

// In a real app, this would filter the data and refresh visualizations

alert('Filters applied! In a real application, this would filter the data and refresh all visualizations.');

}

function resetFilters() {

document.getElementById('time-period').value = 'all';

document.getElementById('entity-type').value = 'all';

document.getElementById('min-amount').value = '10000';

alert('Filters reset! In a real application, this would reset all visualizations to their default state.');

}

function initNetworkGraph() {

const width = document.getElementById('network-graph').clientWidth;

const height = document.getElementById('network-graph').clientHeight;

// Create SVG

const svg = d3.select("#network-graph")

.append("svg")

.attr("width", width)

.attr("height", height);

// Create a group for the graph

const g = svg.append("g");

// Color scale for communities

const color = d3.scaleOrdinal()

.domain([0, 1, 2, 3])

.range(["#e74c3c", "#f39c12", "#2ecc71", "#3498db"]);

// Simulation setup

const simulation = d3.forceSimulation(sampleData.nodes)

.force("link", d3.forceLink(sampleData.links).id(d => d.id).distance(100))

.force("charge", d3.forceManyBody().strength(-300))

.force("center", d3.forceCenter(width / 2, height / 2))

.force("collision", d3.forceCollide().radius(d => d.size / 2 + 5));

// Create links

const link = g.append("g")

.attr("class", "links")

.selectAll("line")

.data(sampleData.links)

.enter().append("line")

.attr("stroke-width", d => Math.sqrt(d.value))

.attr("stroke", "#999")

.attr("stroke-opacity", 0.6);

// Create nodes

const node = g.append("g")

.attr("class", "nodes")

.selectAll("circle")

.data(sampleData.nodes)

.enter().append("circle")

.attr("r", d => d.size / 2)

.attr("fill", d => color(d.community))

.attr("stroke", "#fff")

.attr("stroke-width", 2)

.call(d3.drag()

.on("start", dragstarted)

.on("drag", dragged)

.on("end", dragended));

// Add labels

const label = g.append("g")

.attr("class", "labels")

.selectAll("text")

.data(sampleData.nodes)

.enter().append("text")

.attr("dy", -10)

.text(d => {

if (d.type === 'hospital') return d.name.split(' ')[0];

if (d.type === 'doctor') return d.name.split(' ')[1];

return d.name.split(' ')[0];

})

.attr("font-size", "10px")

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

// Tooltip

const tooltip = d3.select("#graph-tooltip");

node.on("mouseover", function(event, d) {

tooltip.transition()

.duration(200)

.style("opacity", .9);

tooltip.html(`

<strong>${d.name}</strong><br/>

Type: ${d.type}<br/>

Community: ${d.community + 1}<br/>

Claims: ${d.claims}<br/>

Billed: $${d.amount.toLocaleString()}<br/>

Risk: ${Math.round(d.risk \* 100)}%

`)

.style("left", (event.pageX + 10) + "px")

.style("top", (event.pageY - 28) + "px");

// Highlight connected nodes

link.attr("stroke", l => l.source === d || l.target === d ? "#e74c3c" : "#999")

.attr("stroke-width", l => l.source === d || l.target === d ? Math.sqrt(l.value) \* 1.5 : Math.sqrt(l.value));

node.attr("opacity", n => isConnected(d, n) ? 1 : 0.2);

label.attr("opacity", n => isConnected(d, n) ? 1 : 0.2);

});

node.on("mouseout", function() {

tooltip.transition()

.duration(500)

.style("opacity", 0);

link.attr("stroke", "#999")

.attr("stroke-width", d => Math.sqrt(d.value));

node.attr("opacity", 1);

label.attr("opacity", 1);

});

node.on("click", function(event, d) {

updateCommunityDetails(d.community);

});

// Update positions on simulation tick

simulation.on("tick", () => {

link

.attr("x1", d => d.source.x)

.attr("y1", d => d.source.y)

.attr("x2", d => d.target.x)

.attr("y2", d => d.target.y);

node

.attr("cx", d => d.x)

.attr("cy", d => d.y);

label

.attr("x", d => d.x)

.attr("y", d => d.y);

});

// Helper function to check if nodes are connected

function isConnected(a, b) {

return sampleData.links.some(link =>

(link.source === a && link.target === b) ||

(link.source === b && link.target === a)

);

}

// Drag functions

function dragstarted(event, d) {

if (!event.active) simulation.alphaTarget(0.3).restart();

d.fx = d.x;

d.fy = d.y;

}

function dragged(event, d) {

d.fx = event.x;

d.fy = event.y;

}

function dragended(event, d) {

if (!event.active) simulation.alphaTarget(0);

d.fx = null;

d.fy = null;

}

}

function updateCommunityDetails(communityId) {

const community = sampleData.communities.find(c => c.id === communityId);

if (!community) return;

document.getElementById('community-details').style.display = 'none';

document.getElementById('community-metrics').style.display = 'block';

// Update metrics

document.getElementById('collusion-prob').textContent = community.risk;

document.getElementById('collusion-prob').className = `risk-indicator ${

community.risk === 'high' ? 'high-risk' :

community.risk === 'medium' ? 'medium-risk' : 'low-risk'

}`;

document.getElementById('community-size').textContent = community.members.length;

const totalBilled = community.members.reduce((sum, memberId) => {

const member = sampleData.nodes.find(n => n.id === memberId);

return sum + (member ? member.amount : 0);

}, 0);

document.getElementById('total-billed').textContent = totalBilled.toLocaleString();

// Update table

updateCommunityTable(communityId);

}

function initBillingChart() {

const chart = echarts.init(document.getElementById('billing-chart'));

const option = {

title: {

text: 'Billing Patterns by Community',

subtext: 'Monthly billed amounts over the last year',

left: 'center'

},

tooltip: {

trigger: 'axis',

axisPointer: {

type: 'shadow'

}

},

legend: {

data: ['Community 1 (High Risk)', 'Community 2 (Medium Risk)', 'Community 3 (Low Risk)', 'Community 4 (Low Risk)'],

top: 30

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true

},

xAxis: {

type: 'category',

data: sampleData.temporalData.months,

axisLabel: {

rotate: 45

}

},

yAxis: {

type: 'value',

name: 'Billed Amount ($)',

axisLabel: {

formatter: '${value}'

}

},

series: sampleData.communities.map(community => ({

name: `${community.name} (${community.risk} risk)`,

type: 'line',

data: community.billingPattern,

smooth: true,

lineStyle: {

width: 3

},

symbolSize: 8

}))

};

chart.setOption(option);

window.addEventListener('resize', function() {

chart.resize();

});

}

function initReferralChart() {

const chart = echarts.init(document.getElementById('referral-chart'));

// Process data for sunburst

const processReferralData = (communityId) => {

const community = sampleData.communities.find(c => c.id === communityId);

if (!community) return null;

const data = [];

const addedNodes = new Set();

// Add hospitals

community.members.filter(id => id.startsWith('h')).forEach(hospitalId => {

const hospital = sampleData.nodes.find(n => n.id === hospitalId);

if (hospital) {

data.push({

name: hospital.name,

value: hospital.amount,

itemStyle: { color: '#e74c3c' },

children: []

});

addedNodes.add(hospitalId);

}

});

// Add doctors

community.members.filter(id => id.startsWith('d')).forEach(doctorId => {

const doctor = sampleData.nodes.find(n => n.id === doctorId);

if (doctor) {

// Find the hospital this doctor is connected to

const hospitalLink = sampleData.links.find(l =>

l.target === doctorId && l.type === 'referral' && addedNodes.has(l.source));

if (hospitalLink) {

const hospitalNode = data.find(n => n.name === sampleData.nodes.find(n => n.id === hospitalLink.source).name);

if (hospitalNode) {

hospitalNode.children.push({

name: doctor.name,

value: doctor.amount,

itemStyle: { color: '#f39c12' },

children: []

});

addedNodes.add(doctorId);

}

}

}

});

// Add members

community.members.filter(id => id.startsWith('m')).forEach(memberId => {

const member = sampleData.nodes.find(n => n.id === memberId);

if (member) {

// Find the doctor this member is connected to

const doctorLink = sampleData.links.find(l =>

l.target === memberId && l.type === 'claim' && addedNodes.has(l.source));

if (doctorLink) {

// Find the hospital node first

const hospitalNode = data.find(n =>

n.children.some(child =>

child.name === sampleData.nodes.find(n => n.id === doctorLink.source).name));

if (hospitalNode) {

const doctorNode = hospitalNode.children.find(child =>

child.name === sampleData.nodes.find(n => n.id === doctorLink.source).name);

if (doctorNode) {

doctorNode.children.push({

name: member.name,

value: member.amount,

itemStyle: { color: '#3498db' }

});

addedNodes.add(memberId);

}

}

}

}

});

return data;

};

// Use community 0 (high risk) for the initial view

const data = processReferralData(0);

const option = {

title: {

text: 'Referral Pattern for Community 1 (High Risk)',

subtext: 'Hospital → Doctor → Member Association',

left: 'center'

},

series: {

name: 'Referral Flow',

type: 'sunburst',

data: data,

radius: [0, '90%'],

label: {

rotate: 'radial'

},

levels: [

{},

{

r0: '15%',

r: '45%',

itemStyle: {

borderWidth: 2

},

label: {

rotate: 'tangential'

}

},

{

r0: '45%',

r: '80%',

label: {

align: 'right'

}

},

{

r0: '80%',

r: '90%',

label: {

position: 'outside',

padding: 3,

silent: false

},

itemStyle: {

borderWidth: 3

}

}

]

},

tooltip: {

trigger: 'item',

formatter: params => {

const name = params.name;

const value = params.value.toLocaleString();

return `<strong>${name}</strong><br/>Billed Amount: $${value}`;

}

}

};

chart.setOption(option);

window.addEventListener('resize', function() {

chart.resize();

});

}

function initMap() {

const map = L.map('map').setView([37.8, -96], 4);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

}).addTo(map);

// Add markers for each community with different colors

const communityColors = {

0: '#e74c3c',

1: '#f39c12',

2: '#2ecc71',

3: '#3498db'

};

sampleData.communities.forEach(community => {

const communityNodes = sampleData.nodes.filter(node =>

community.members.includes(node.id));

// Create a feature group for this community

const group = L.featureGroup();

communityNodes.forEach(node => {

const marker = L.circleMarker([node.lat, node.lng], {

radius: node.size / 5,

fillColor: communityColors[community.id],

color: '#fff',

weight: 1,

opacity: 1,

fillOpacity: 0.8

}).bindPopup(`

<b>${node.name}</b><br>

Type: ${node.type}<br>

Community: ${community.name}<br>

Risk: ${community.risk}<br>

Claims: ${node.claims}<br>

Billed: $${node.amount.toLocaleString()}

`);

group.addLayer(marker);

});

// Add lines between connected nodes

community.members.forEach(sourceId => {

community.members.forEach(targetId => {

if (sourceId < targetId) { // Avoid duplicate links

const link = sampleData.links.find(l =>

(l.source === sourceId && l.target === targetId) ||

(l.source === targetId && l.target === sourceId));

if (link) {

const sourceNode = sampleData.nodes.find(n => n.id === sourceId);

const targetNode = sampleData.nodes.find(n => n.id === targetId);

if (sourceNode && targetNode) {

const line = L.polyline(

[[sourceNode.lat, sourceNode.lng], [targetNode.lat, targetNode.lng]],

{

color: communityColors[community.id],

weight: Math.sqrt(link.value) / 2,

opacity: 0.7

}

);

group.addLayer(line);

}

}

}

});

});

group.addTo(map);

// Add legend for this community

const legend = L.control({position: 'bottomright'});

legend.onAdd = function(map) {

const div = L.DomUtil.create('div', 'info legend');

div.innerHTML += `

<div style="background:${communityColors[community.id]}; width:20px; height:20px; display:inline-block;"></div>

${community.name} (${community.risk} risk)<br>

`;

return div;

};

legend.addTo(map);

});

// Fit bounds to show all markers

const markers = [];

sampleData.nodes.forEach(node => {

markers.push([node.lat, node.lng]);

});

map.fitBounds(markers);

}

function initRiskChart() {

const ctx = document.getElementById('risk-chart').getContext('2d');

// Prepare data

const labels = sampleData.communities.map(c => c.name);

const riskScores = sampleData.communities.map(c => {

const members = sampleData.nodes.filter(n => c.members.includes(n.id));

const avgRisk = members.reduce((sum, m) => sum + m.risk, 0) / members.length;

return avgRisk \* 100;

});

const billedAmounts = sampleData.communities.map(c => {

const members = sampleData.nodes.filter(n => c.members.includes(n.id));

return members.reduce((sum, m) => sum + m.amount, 0) / 1000000; // in millions

});

new Chart(ctx, {

type: 'bar',

data: {

labels: labels,

datasets: [

{

label: 'Risk Score (%)',

data: riskScores,

backgroundColor: ['#e74c3c', '#f39c12', '#2ecc71', '#3498db'],

borderColor: ['#c0392b', '#d35400', '#27ae60', '#2980b9'],

borderWidth: 1,

yAxisID: 'y'

},

{

label: 'Billed Amount ($M)',

data: billedAmounts,

type: 'line',

borderColor: '#2c3e50',

borderWidth: 2,

fill: false,

yAxisID: 'y1'

}

]

},

options: {

responsive: true,

plugins: {

title: {

display: true,

text: 'Risk Score vs Billed Amount by Community'

},

tooltip: {

callbacks: {

label: function(context) {

let label = context.dataset.label || '';

if (label) {

label += ': ';

}

if (context.datasetIndex === 0) {

label += context.raw.toFixed(1) + '%';

} else {

label += '$' + context.raw.toFixed(2) + 'M';

}

return label;

}

}

}

},

scales: {

y: {

type: 'linear',

display: true,

position: 'left',

title: {

display: true,

text: 'Risk Score (%)'

},

max: 100

},

y1: {

type: 'linear',

display: true,

position: 'right',

title: {

display: true,

text: 'Billed Amount ($M)'

},

grid: {

drawOnChartArea: false

}

}

}

}

});

}

function initProcedureChart() {

const chart = echarts.init(document.getElementById('procedure-chart'));

// Prepare data - use community 0 (high risk) as example

const community = sampleData.communities[0];

const data = community.procedureCodes.map(pc => ({

name: pc.code,

value: pc.count,

avgAmount: pc.avgAmount

}));

const option = {

title: {

text: 'Top Procedure Codes for Community 1 (High Risk)',

subtext: 'Frequency and average amount',

left: 'center'

},

tooltip: {

formatter: params => {

return `<strong>${params.name}</strong><br/>

Frequency: ${params.value}<br/>

Avg Amount: $${params.data.avgAmount}`;

}

},

xAxis: {

type: 'category',

data: data.map(item => item.name),

axisLabel: {

rotate: 45

}

},

yAxis: [

{

type: 'value',

name: 'Frequency',

axisLabel: {

formatter: '{value}'

}

},

{

type: 'value',

name: 'Avg Amount ($)',

axisLabel: {

formatter: '${value}'

}

}

],

series: [

{

name: 'Frequency',

type: 'bar',

data: data.map(item => item.value),

itemStyle: {

color: '#3498db'

}

},

{

name: 'Average Amount',

type: 'line',

yAxisIndex: 1,

data: data.map(item => item.avgAmount),

itemStyle: {

color: '#e74c3c'

},

symbolSize: 8,

lineStyle: {

width: 3

}

}

]

};

chart.setOption(option);

window.addEventListener('resize', function() {

chart.resize();

});

}

function initTemporalChart() {

const ctx = document.getElementById('temporal-chart').getContext('2d');

new Chart(ctx, {

type: 'line',

data: {

labels: sampleData.temporalData.months,

datasets: sampleData.temporalData.communityClaims.map((community, idx) => ({

label: community.name,

data: community.data,

borderColor: [

'#e74c3c',

'#f39c12',

'#2ecc71',

'#3498db'

][idx],

backgroundColor: 'rgba(255, 255, 255, 0)',

borderWidth: 2,

tension: 0.3,

pointRadius: 4,

pointHoverRadius: 6

}))

},

options: {

responsive: true,

plugins: {

title: {

display: true,

text: 'Monthly Claim Patterns by Community'

},

tooltip: {

mode: 'index',

intersect: false

}

},

scales: {

y: {

beginAtZero: true,

title: {

display: true,

text: 'Number of Claims'

}

}

},

interaction: {

mode: 'nearest',

axis: 'x',

intersect: false

}

}

});

}

function initCommunityTable() {

// Default to showing all entities

updateCommunityTable(null);

}

function updateCommunityTable(communityId) {

const tbody = document.querySelector('#community-table tbody');

tbody.innerHTML = '';

const entitiesToShow = communityId === null ?

sampleData.nodes :

sampleData.nodes.filter(node =>

sampleData.communities[communityId].members.includes(node.id));

entitiesToShow.forEach(entity => {

const row = document.createElement('tr');

// Calculate referrals

const referrals = sampleData.links.filter(l =>

l.source === entity.id && l.type === 'referral').length;

// Calculate claims

const claims = entity.claims;

row.innerHTML = `

<td>${entity.type.charAt(0).toUpperCase() + entity.type.slice(1)}</td>

<td>${entity.name}</td>

<td>${claims.toLocaleString()}</td>

<td>$${entity.amount.toLocaleString()}</td>

<td>${referrals}</td>

<td><span class="risk-indicator ${

entity.risk > 0.7 ? 'high-risk' :

entity.risk > 0.4 ? 'medium-risk' : 'low-risk'

}">${Math.round(entity.risk \* 100)}%</span></td>

`;

tbody.appendChild(row);

});

}

</script>

</body>

</html>

Key Features of This Implementation

1. **Interactive Network Graph**:
   * Visualizes connections between hospitals, doctors, and member associations
   * Uses Louvain algorithm-inspired community detection (simulated in this example)
   * Color-codes communities by risk level
2. **Multiple Analysis Views**:
   * Billing patterns over time
   * Referral patterns sunburst chart
   * Geographic distribution map
   * Risk score comparison
   * Procedure code analysis
   * Temporal claim patterns
3. **Interactive Features**:
   * Filter controls for time period, entity type, and dollar amounts
   * Tooltips with detailed information
   * Click-to-explore community details
   * Responsive design that works on different screen sizes
4. **Comprehensive Data Display**:
   * Detailed table of community members
   * Risk assessment metrics
   * Anomaly indicators

How to Use This Dashboard

1. **Network Graph**: Click on nodes to see detailed information about communities
2. **Filters**: Use the controls at the top to filter data (simulated in this example)
3. **Charts**: Hover over chart elements to see detailed values
4. **Map**: Click on markers to see entity details
5. **Table**: Scroll through detailed information about all entities

Note: In a real implementation, you would connect this to actual data sources and implement the Louvain algorithm for community detection. This example uses simulated data to demonstrate the visualization concepts.