GSLetterNeo vol.129

2019年4月

Webブラウザ上での複数の2次元情報の重ね合わせ

松原 伸人 matubara@sra.co.jp

はじめに

「カタマリを見つけて辿れるヒストリデータブラウザ(3)」(GSLetterNeo Vol.93) では、歴史データなどをイベントごとに時間軸上に配置して表示することで、イベントが連続的に起きている時間帯や切れ目を見つけて辿れるようにするWebアプリケーションのプロトタイプを紹介しました。

今回は京都の歴史データに位置情報を試験的に付加したデータを例に、Webアプリケーションの地図プログラムなどと2次元平面上に重ねたり、並べたりして表示する方法を紹介します。[図1]

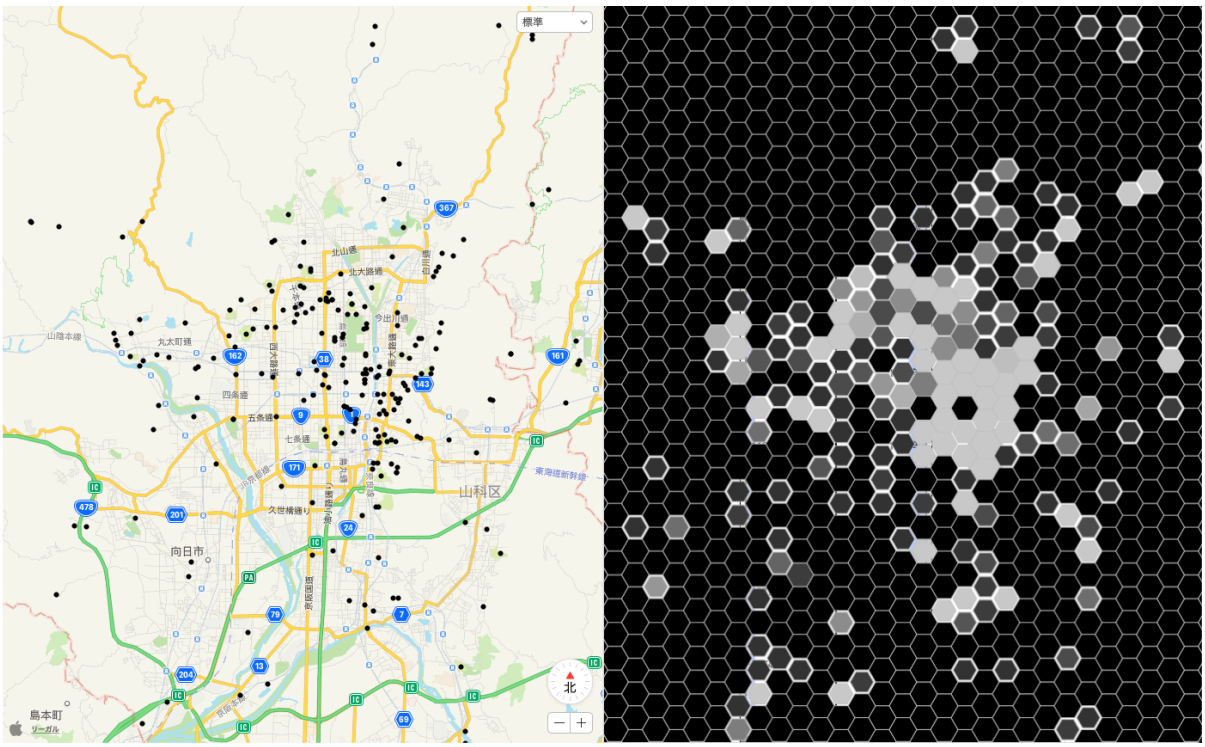

図1 左側 重ねて表示、右側 並べて表示

本内容は、科学技術振興機構(JST) 戦略的創造研究推進事業(CREST) 「データ粒子化による高速高精度な次世代マイニング技術の創出」プロジェクト（研究代表者：宇野毅明 教授 (国立情報学研究所)）で行なっている研究開発の一部です。今回使用しているデータは、北雄介先生 (長岡造形大学) 等により、京都の歴史<10> 年表・事典(1976年) 京都市 学芸書林 から年ごとに抽出した出来事を表すテキスト群と、同索引より抽出した用語とWikipediaから得られた経緯度からなる位置情報を用い、著者自らが位置情報を持つ用語ごとに、用語を含む出来事テキストを抽出したデータを使用しています。地図プログラムは Apple MapKit JS を使用しました。

HTMLエレメントを重ねる

図1の画像左側は地図の上にデータを重ねて表示する例、画像右側は地図とデータを並べて表示する例です。Webアプリケーションの画面レイアウトにはCSS の display grid を用いています。display grid は、画面を縦横にグリッド分割してHTML要素を配置するレイアウト方法です。grid-template プロパティを用い、グリッドのセルの縦方向の列ごとの幅、横方向の行ごとの高さを設定できます。複数のHTMLエレメントを同じセルに配置すると同じ大きさで重ねられます。z-index プロパティを設定するとHTMLエレメントが重なる順番を設定できます。重ねて表示するセルをクラスにして設定しておけば、重ねたいHTMLエレメントにクラス名を書くだけで、同じ場所に大きさを揃えて重ねて表示できます。

HTMLエレメントを並べる

gridレイアウトを用いて横に並べる場合、隣り合う列の幅を同じサイズにし、2つのHTMLエレメントを並べて配置します。position absolute を用いると、HTMLエレメントのtop、right、bottom、leftの位置を、親要素の表示エリアの左上隅を0,0とする座標や、ウインドウの表示エリアの左上隅を0,0とする座標に設定できます。HTMLエレメントをgridにより大きさを揃え、topやleftの位置指定すると、表示エリアの縦横の幅を揃えたまま任意の場所へ配置できます。

データを描く

二条城の位置を中央に表示し、その上に黒い丸を描画するプログラム例を、**位置情報データを描画するプログラム例** に掲載します。図2 はこのプログラムを実行した画面のスクリーンショットです。

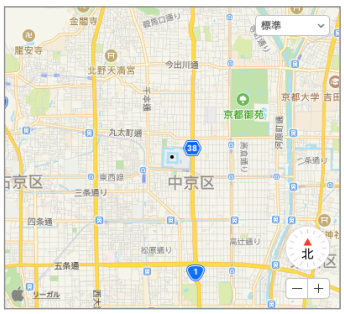

図2 重ねて表示する

HTML の Canvasエレメント を用いると画面上の位置をピクセルで指定して線や円など簡単な図形を描けます。前述のように display grid を用いて設定したセルに Canvasエレメント を配置すれば、複数の Canvasエレメント の位置と大きさを揃えて重ねて用いることができます。Canvasエレメントに描画するエリアの大きさ width と height は、画面上にレイアウトした大きさに設定されず初期値のまま変らないため別途設定します。地図などと重ねて表示する場合、レイアウトした大きさに描画するエリアの大きさを揃えます。HTMLエレメントはページのDOMツリーに加わって表示された状態になると、ページ内における位置と大きさを JavaScript の HTMLエレメントの関数 getBoundingClientRect で得られるようになります。データを描き始める前、ページをロードした後に、Canvasエレメント の大きさを設定します。ウインドウの大きさが変わったりレイアウトが変わるとレイアウトの大きさと描画エリアの大きさが合わなくなります。ウインドウの大きさの変更は window のイベントリスナーを用い、ウインドウの大きさの変更時に resizeCanvas を実行するように設定します。

地図上の位置と同じ座標に黒い丸を描く場合、経緯度を表示エリアの座標に変換して canvas の arc 関数に変換した座標を指定して円を描きます。経緯度から表示エリアへの座標変換は、Apple MapKit JS を使う場合は経緯度を表す Coordinate の関数 toMapPointを用います。toMapPoint関数は、経緯度Coodinateを横軸x縦軸yの直交座標 MapPoint に変換します。地図で表示している領域は Map.region プロパティで得られます。MapRect.toMapRect関数 により MapPoint と同様の直交座標で表す矩形が得られます。前述したように Canvas の大きさは getBoundingClientRect で得られます。MapPoint を表示エリアの座標に変換するプログラムは、記述例の mapPointToViewPoint のように書けます。地図を回転して向きを変えている場合は、地図の向きを Map.rotation プロパティから得た角度で回転変換すれば同様の方法を適用できます。

次のように style の .canvas に left: 50vw を加えると、地図を左、Canvasを右に配置して並べて表示し

```
.canvas {
    z-index: 2;
    position: absolute;
    left: 50vw;
}
```

ます。vwは表示エリアの横幅に対する比率を表す単位です。100vwの時に横幅と等しくなり50vwで横幅の半分の大きさになります。leftプロパティに設定しているので、横方向で画面中央の位置に Canvas の左端がきます。[図3]

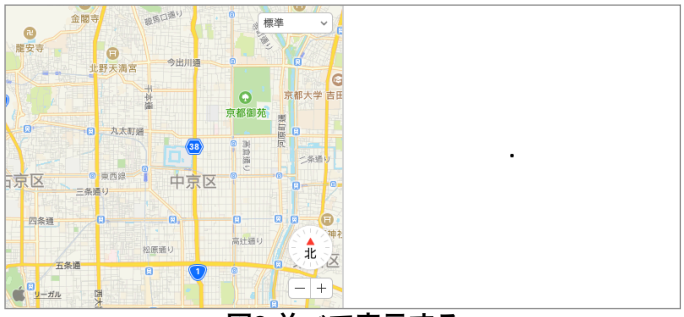

図3 並べて表示する

位置情報データを描画するプログラム例

```
<html><head>
<style>
body {
	display: grid;
	grid-template-columns: 50% 50%;
	grid-template-rows: 100%;
	width: 100%;
	height: 100%;
	margin: 0;
	position: relative;
}
.layer {
	display: block;
	grid-column: 1/2;
	grid-row: 1/2;
	width: 100%;
	height: 100%;
	position: absolute;
}
.map {
	z-index: 1;
}
.canvas {
	z-index: 2;
}
</style>
<script src="https://cdn.apple-mapkit.com/mk/5.x.x/mapkit.js"></script>
<script>
mapkit.init({
	authorizationCallback: (done) => {...}
})
function initCanvas() {
	document.body.appendChild(canvas)
	resizeCanvas()
}
function resizeCanvas() {
	let rect = canvas.getBoundingClientRect()
	canvas.width = rect.width
	canvas.height = rect.height
	draw()
}
function draw() {
	let ctx = canvas.getContext("2d"),
		rect = canvas.getBoundingClientRect()
	ctx.clearRect(0, 0, rect.width, rect.height)
	drawLocationOn(canvas.getContext("2d"), nijojoLocation)
}
function drawLocationOn (ctx, location) {
	let point = mapPointToViewPoint(new mapkit.Coordinate(location.lat, location.lng).toMapPoint(), map.region.toMapRect(),
canvas.getBoundingClientRect()),
		radius = 2
	ctx.fillStyle = "black"
	ctx.beginPath()
	ctx.arc(point.x, point.y, radius, 0, Math.PI * 2, false)
	ctx.fill()
}
function mapPointToViewPoint(mapPoint, mapRect, viewRect) {
	return {
		x: (mapPoint.x - mapRect.origin.x) / mapRect.size.width * viewRect.width,
		y: (mapPoint.y - mapRect.origin.y) / mapRect.size.height * viewRect.height
	}
}
function initMap() {
	document.body.appendChild(mapContainer)
	map.region = new mapkit.CoordinateRegion(
		new mapkit.Coordinate(nijojoLocation.lat, nijojoLocation.lng),
		new mapkit.CoordinateSpan(0.10, 0.10))
}
mapkit.addEventListener("configuration-change", (evt) => {
	switch (evt.status) {
		case "Initialized": draw(); break;
	}
})
window.addEventListener('resize', () => {
	resizeCanvas()
	draw()
})
let canvas = document.createElement("canvas"),
	mapContainer = document.createElement("div"),
	map = new mapkit.Map(mapContainer),
	nijojoLocation = {
		lat: 35.0142,
		lng: 135.7475
	}
canvas.className = "canvas layer"
mapContainer.className = "map layer"
</script>
</head><body onload="initMap(); initCanvas()"></body></html>
```

表示例

冒頭に書いたように、京都の歴史<10>の年表から年ごとに出来事を抽出し、索引とwikipediaで得られた経緯度をもとにして作成したデータを表示してみます。

年表から得られた出来事の数は 12,000 件、索引から得られた用語の数は 16,768 語、そのうち wikipedia から経緯度が得られた場所を表す用語は 501 語でした。1個の出来事の中に複数の場所が出てきたり、1個の場所が様々な出来事の中に現れています。

冒頭に掲載した図1の左側は、京都市周辺で、出来事が現れる場所を黒い丸で表しています。プログラム例で二条城の位置に黒丸を描くのと同様に、drawLocationOn関数に経緯度を指定して複数の場所に黒丸を描画します。経緯度の配列locationsを描画するコードの場合次のように書けます。

```
locations.forEach((location) => {
    drawLocationOn(ctx, location)
})
```

図1の右側は、京都市周辺のエリアを、対角線が1kmの六角形エリアに分割し、六角形エリアごとに現れる場所の数が多いほど六角形を大きく、少ないと小さく表示しています。出来事が多く記録されているエリアは濃い灰色、少ないエリアは薄い灰色で表しています。黒色の六角形エリアはデータが無い場所を表します。

六角形のエリアは頂点を経緯度で表すデータを作成しています。501個の位置情報がどの六角形エリアにあるかを調べて集計できます。点が六角形の中にあるか判定する方法は、点と多角形の内外判定などで検索すると様々な方法が見つけられます。

図5,6は、年ごとに出来事が現れるエリアを円で表しています。同じ年に現れる出来事をグループとし

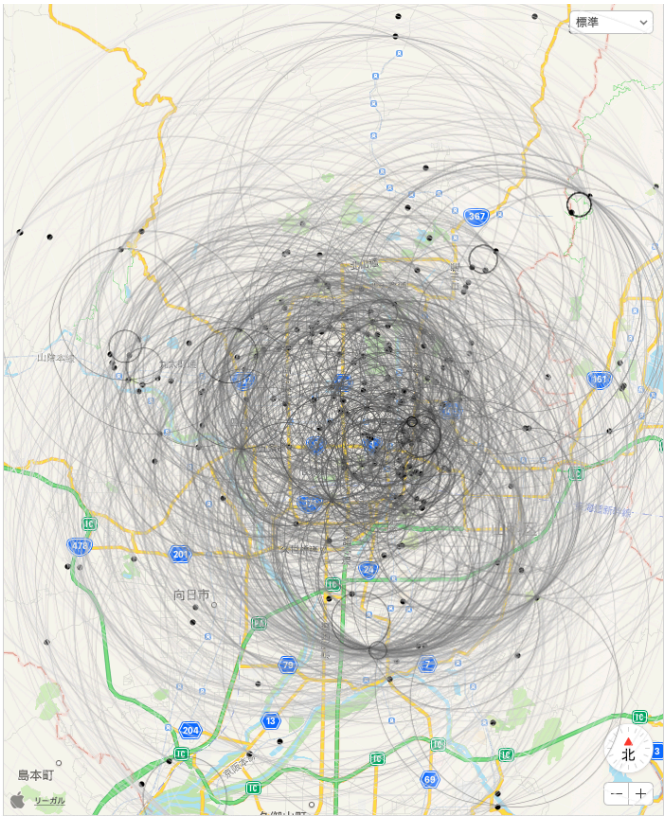

図5 年毎に出来事が現れるエリアを円で表す例 (重ねて表示)

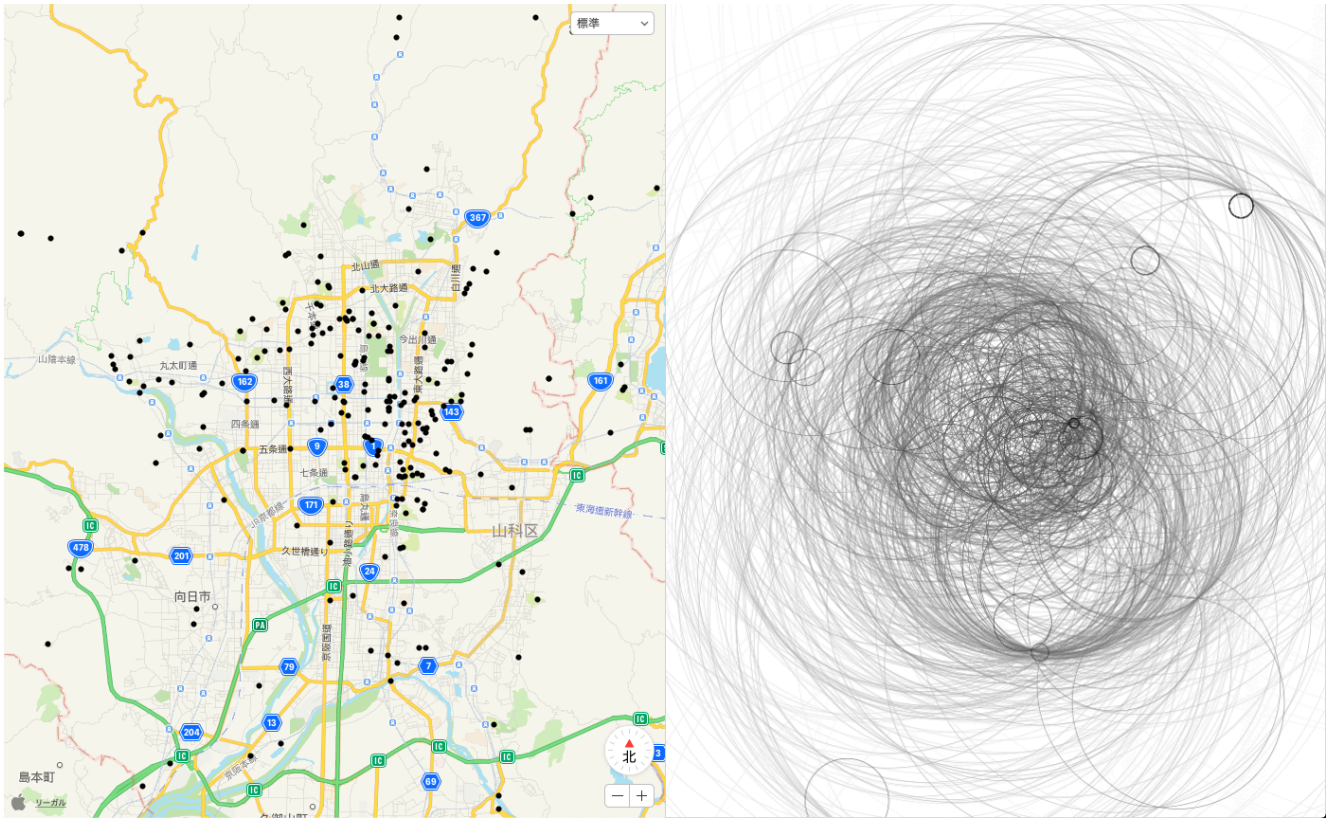

図6 年毎に出来事が現れるエリアを円で表す例 (並べて表示)

て、グループごとに出来事が現れる場所を内包する円を求めて描いています。大きな円の色は薄く下に描き、小さな円は濃く上に描いています。

GSLetterNeo Vol.126 「距離が近いデータをグループにして表示する」で紹介した方法を用いることで、同じ年に現れる出来事によるグループ化だけでなく、出来事が起こる年の期間が近いデータをグループにしたりできます。図7は、出来事の記述がない期間で区切って分けた15グループを円で表しています。

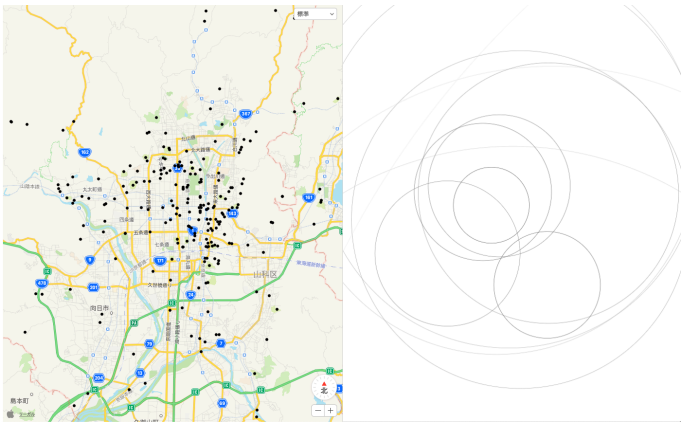

図7 15グループのエリア

15グループの期間と現れる出来事の数は、570年 1件、603年 1件、623年 2件、698 – 702年 3件、711年 1件、781 – 784年 2件、793 – 812年 15件、817 – 827年 13件、836 – 841年 5件、847 – 851年 2件、856 – 866年 11件、873 – 897年 23件、902 – 976年 62件、985 – 1033年 52件、1038 – 1971年 2822件でした。

図8は、出来事ごとに記載されている複数の場所を直線で結んで描いた例です。線が無い場所は出来事の中に記述されている場所が1箇所あるいは同じ場所を指していることを表しています。多くの出来事の中に現れる場所にはたくさんの線が集まっています。

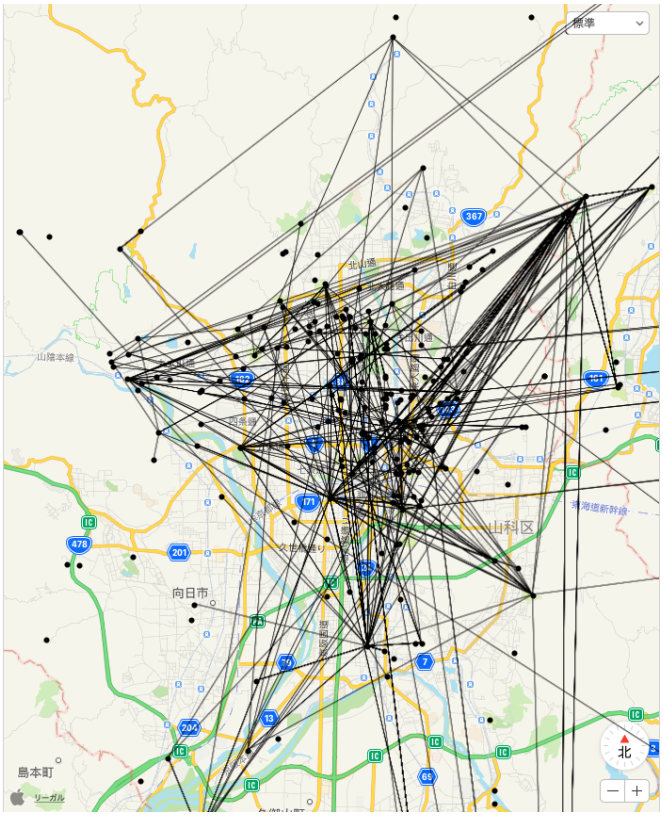

図8 出来事が示す場所を直線で結んで描く例

参考

以下は今回紹介したプログラムで用いた主なAPIのドキュメントへのリンクです。

grid - CSS/ カスケーディングスタイルシート | MDN

MapKit JS - Maps - Apple Developer

mapkit.Coordinate - mapkit | Apple Developer Documentation

mapkit.MapPoint - mapkit | Apple Developer Documentation

mapkit.MapRect - mapkit | Apple Developer Documentation

Element.getBoundingClientRect() - Web API | MDN

CanvasRenderingContext2D.arc() - Web API | MDN

GSLetterNeo vol.129
発行日 2019年4月20日
発行者 株式会社 ＳＲＡ 先端技術研究所
編集者 土屋 正人
バックナンバー https://www.sra.co.jp/gsletter/

お問い合わせ
gsneo@sra.co.jp
〒171-8513 東京都豊島区南池袋2-32-8