

Suivi de la population des loups en Suisse

L. Dagneau, N. Trojan

Janvier 2025, Université de Lausanne

<u>1. Introduction et contexte</u>	2
<u>2. Collecte de données</u>	2
<u>3. Méthodologie</u>	3
3.1 QGIS	3
3.2 HTML	3
3.3 JAVASCRIPT	7
3.4 CSS	25
<u>4. Résultats</u>	29
<u>5. Limites de l'application</u>	31
<u>6. Conclusion</u>	32
<u>7. Références</u>	32

1. Introduction et contexte

Depuis 2000, la population de loups en Suisse a connu une croissance significative. Après une réapparition spontanée dans les années 1990, la première meute stable s'est formée en 2012 dans le canton des Grisons. Le nombre de meutes est passé de 11 en 2020 à 32 en 2023, avec environ 300 individus, entraînant une augmentation notable des cas de préddation sur le bétail, passant de 446 incidents en 2019 à 1 480 en 2022 (OFEV, 2023). Pour répondre à cette situation, la Suisse a adopté des mesures de gestion, dont une modification de la loi sur la chasse en décembre 2022, autorisant depuis décembre 2023 l'abattage préventif des loups afin de limiter les dégâts futurs (Pro Natura s. d.). Les résultats observés après l'entrée en vigueur de cette loi montrent une baisse de la préddation dans certains cantons : -20 % en Valais, -35 % dans les Grisons, et des baisses moins marquées dans d'autres régions comme le Tessin (-34 %) et Saint-Gall (+18 %). En revanche, le canton de Vaud a enregistré une légère hausse (+3 %) (RSI 2023).

Nous avons choisi ce thème car il représente un sujet sensible à plusieurs niveaux, qu'il s'agisse des enjeux liés à la conservation de la nature, à la protection des activités agricoles ou à la gestion de la faune. Ce sujet nous intéresse particulièrement, car il touche des domaines cruciaux pour l'avenir de l'environnement et de l'agriculture, et soulève des questions complexes concernant la coexistence entre l'homme et la faune sauvage. Ainsi, la Suisse s'efforce de trouver un équilibre entre la conservation du loup et la protection des activités agricoles, en mettant en place des stratégies qui combinent mesures préventives et interventions ciblées, afin de garantir une coexistence durable entre l'homme et ce grand prédateur (OFEV, 2023).

Dans ce contexte, nous avons développé une carte web interactive qui permet de suivre les observations et les déplacements des loups sur le territoire suisse. Cet outil permet d'explorer les données de manière dynamique, en affichant des informations spatiales et temporelles sur les déplacements des loups. Grâce à cette carte, il est possible d'analyser non seulement l'évolution temporelle des observations, mais aussi leur distribution géographique, dans le but de fournir une analyse détaillée et accessible. Nous espérons que cette initiative contribuera à une meilleure compréhension du phénomène et fournira des données utiles pour soutenir des stratégies équilibrées entre la conservation et la gestion de la faune sauvage.

2. Collecte de données

Les données utilisées pour le projet peuvent être divisées en deux grandes catégories. D'une part, nous disposons de données sur les observations de loups en Suisse. Ces données sont fournies par l'Institut pour l'écologie des carnivores et la gestion de la faune sauvage (KORA) et font état des observations par reconnaissance génétique avec le lieu, la date de l'observation et l'ID du loup. La deuxième catégorie de données sont les données géographiques, afin de géoréférencer les observations et de les traduire sur une carte. Ces données, issues du dataset swissTLMRegio, ont été téléchargées à partir du catalogue en ligne de Swisstopo.

3. Méthodologie

Dans cette section, la méthodologie de création de la carte sera abordée en 4 étapes : la partie sur QGIS d'abord et ensuite les trois codes html, javascript et css.

3.1 QGIS

Dans un premier temps, les données sur les observations de loups ont été géoréférencées avec le fichier shape du territoire suisse. Pour ce faire, une liaison par attribut a été effectuée. Les observations contenaient le nom du hameau de la municipalité, qui était absent du fichier Swisstopo, et ont été éditées manuellement pour correspondre. En outre, le fichier des observations était au format PDF et ne pouvait pas être formaté directement en tant que tableau, de sorte que les données ont été en grande partie transcrrites manuellement. À partir de cette liaison, nous avons obtenu le fichier de forme de toutes les observations. La deuxième cuoche créée est celle des mouvements des loups, obtenue avec la fonction « points to lines » et incluant la date de l'observation dans un format adéquat afin de pouvoir ensuite créer une animation du mouvement dans l'ordre chronologique. Pour conclure la partie sur QGIS, les deux couches ont été exportées au format geoJSON pour assurer une meilleure compatibilité avec le script java.

3.2 HTML

Dans la première section du HTML (*head*), nous avons défini la structure de base de la page web. Le contenu des paramètres clés comme le titre de la page web, le jeu de caractères UTF-8 et pour garantir la mise en page responsive. Nous avons intégré les bibliothèques Leaflet et MarkerCluster pour les fonctionnalités de la carte ainsi qu'une feuille de style pour l'apparence de la carte.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mouvements de loups sur le territoire suisse</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Analyse spatiale et temporelle des observations des loups en Suisse entre 1999 et 2022.">
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet.markercluster/1.5.3/MarkerCluster.css" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet.markercluster/1.5.3/MarkerCluster.Default.css" />
    <link rel="stylesheet" href=".//css/style.css" />
  </head>
```

Dans la section *body* nous avons structuré le contenu de la page web en intégrant plusieurs éléments interactifs. Un en-tête *header* présente les titres. Une section spécifique aux sources de données (*data-sources*) offre des liens vers les site web ayant fourni les informations utilisées.

Ensuite, plusieurs conteneurs (*<div>*) sont mis en place pour héberger les éléments de la carte et des graphiques. Des filtres sont également intégrés, permettant de restreindre les données visibles par genre, année, canton ou ID spécifique et sont interconnectés avec la fonction *applyFilters()*. Un bouton de réinitialisation des filtres est aussi ajouté pour revenir rapidement à la vue initiale.

Enfin, les scripts et bibliothèques nécessaires au bon fonctionnement de l'application sont intégrés à la fin de cette section. Cela inclut *Leaflet* et *Leaflet.MarkerCluster* pour la carte, *D3.js* pour les graphiques interactifs, et *Turf.js* pour les calculs de chemins et le script javascript.

```
<body>
    <!--Titre-->
    <div id="header">
        <h1>Suivi de la population de loups en Suisse</h1>
        <h2>Analyse spatiale et temporelle des observations entre 1999 et 2022 sur la base de la trace
    par reconnaissance génétique fournie par l'insitut KORA </h2>
    </div>

    <!--Sources-->
    <div id="data-sources">
        <p>Sources de données: <a href="https://www.kora.ch/fr" target="_blank">KORA</a> et <a
    href="https://www.swisstopo.admin.ch/de/landschaftsmodell-swissutmregio" target="_blank">Swisstopo</a></p>
    </div>

    <!--Conteneurs-->
    <div id="map"></div>
    <div id="year-chart" class="chart"></div>
    <div id="canton-chart" class="chart"></div>
    <div id="scatter-plot" class="chart"></div>
    <div id="time-slider-container">
        <input type="range" id="time-slider" min="1999" max="2022" step="1" value="1999" />
        <label id="year-label">1999</label>
    </div>
    <div id="filters">
        <!--Filtre genre-->
        <label for="filter-gender">Genre:</label>
        <select id="filter-gender" onchange="applyFilters()">
            <option value="">Tous</option>
            <option value="M">Mâles</option>
            <option value="F">Femelles</option>
        </select>

        <!--Filtre années-->
        <label for="filter-year">Année:</label>
        <select id="filter-year" onchange="applyFilters()">
            <option value="">Tous</option>
            <option value="1999">1999</option>
            <option value="2000">2000</option>
            <option value="2001">2001</option>
            <option value="2002">2002</option>
            <option value="2003">2003</option>
            <option value="2004">2004</option>
            <option value="2005">2005</option>
            <option value="2006">2006</option>
            <option value="2007">2007</option>
            <option value="2008">2008</option>
        </select>
    </div>
</body>
```

```

<option value="2009">2009</option>
<option value="2010">2010</option>
<option value="2011">2011</option>
<option value="2012">2012</option>
<option value="2013">2013</option>
<option value="2014">2014</option>
<option value="2015">2015</option>
<option value="2016">2016</option>
<option value="2017">2017</option>
<option value="2018">2018</option>
<option value="2019">2019</option>
<option value="2020">2010</option>
<option value="2021">2021</option>
<option value="2022">2022</option>
</select>

<!--Filtre Canton-->
<label for="filter-canton">Canton:</label>
<select id="filter-canton" onchange="applyFilters()">
    <option value="">Tous</option>
    <option value="Aargau">Aargau</option>
    <option value="Appenzell Ausserrhoden">Appenzell Ausserrhoden</option>
    <option value="Bern">Bern</option>
    <option value="Fribourg">Fribourg</option>
    <option value="Glarus">Glarus</option>
    <option value="Graubünden">Graubünden</option>
    <option value="Luzern">Luzern</option>
    <option value="Nidwalden">Nidwalden</option>
    <option value="Obwalden">Obwalden</option>
    <option value="Schwyz">Schwyz</option>
    <option value="St- Gallen">St- Gallen</option>
    <option value="Thurgau">Thurgau</option>
    <option value="Ticino">Ticino</option>
    <option value="Uri">Uri</option>
    <option value="Vaud">Vaud</option>
    <option value="Valais">Valais</option>
    <option value="Zürich">Zürich</option>
</select>

<!--Filtre ID unique-->
<label for="filter-wolf-id">ID:</label>
<select id="filter-wolf-id" onchange="applyFilters()">
    <option value="">Tous</option>
    <option value="F01">F01</option>
    <option value="F05">F05</option>
    <option value="F07">F07</option>
    <option value="F11">F11</option>
    <option value="F14">F14</option>
    <option value="F16">F16</option>
    <option value="F18">F18</option>
    <option value="F19">F19</option>
    <option value="F24">F24</option>
    <option value="F28">F28</option>
    <option value="F31">F31</option>
    <option value="F32">F32</option>
    <option value="F33">F33</option>
    <option value="F35">F35</option>
    <option value="F37">F37</option>
    <option value="F38">F38</option>
    <option value="F43">F43</option>

```

```
<option value="F45">F45</option>
<option value="F49">F49</option>
<option value="F57">F57</option>
<option value="F59">F59</option>
<option value="F60">F60</option>
<option value="F64">F64</option>
<option value="F75">F75</option>
<option value="F78">F78</option>
<option value="F94">F94</option>
<option value="M03">M03</option>
<option value="M09">M09</option>
<option value="M103">M103</option>
<option value="M107">M107</option>
<option value="M108">M108</option>
<option value="M109">M109</option>
<option value="M11">M11</option>
<option value="M131">M131</option>
<option value="M133">M133</option>
<option value="M135">M135</option>
<option value="M153">M153</option>
<option value="M157">M157</option>
<option value="M159">M159</option>
<option value="M16">M16</option>
<option value="M162">M162</option>
<option value="M169">M169</option>
<option value="M172">M172</option>
<option value="M182">M182</option>
<option value="M186">M186</option>
<option value="M187">M187</option>
<option value="M189">M189</option>
<option value="M190">M190</option>
<option value="M20">M20</option>
<option value="M243">M243</option>
<option value="M28">M28</option>
<option value="M30">M30</option>
<option value="M32">M32</option>
<option value="M34">M34</option>
<option value="M35">M35</option>
<option value="M36">M36</option>
<option value="M38">M38</option>
<option value="M43">M43</option>
<option value="M45">M45</option>
<option value="M46">M46</option>
<option value="M47">M47</option>
<option value="M51">M51</option>
<option value="M52">M52</option>
<option value="M56">M56</option>
<option value="M59">M59</option>
<option value="M64">M64</option>
<option value="M68">M68</option>
<option value="M71">M71</option>
<option value="M73">M73</option>
<option value="M74">M74</option>
<option value="M75">M75</option>
<option value="M76">M76</option>
<option value="M82">M82</option>
<option value="M92">M92</option>
<option value="M95">M95</option>
<option value="M97">M97</option>
</select>
```

```

</select>

<!-- Buttons -->
<button id="reset-filters" onclick="resetFilters()">Reset</button>

<!--Libraries, scripts et plugins-->
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet.markercluster/1.5.3/
leaflet.markercluster.js"></script>
<script src=".js/polylinedecorator.js"></script>
<script src="https://d3js.org/d3.v7.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@turf/turf/turf.min.js"></script>
<script src="js/script.js" defer></script>

</body>
</html>

```

3.3 JAVASCRIPT

Dans cette première section de JavaScript, nous avons initialisé la carte interactive en utilisant la bibliothèque Leaflet.js. Les limites de la Suisse sont définies via le tableau *switzerlandBounds*. Ces coordonnées garantissent que la navigation dans la carte reste centrée sur la région d'étude en empêchant les utilisateurs de dézoomer ou de se déplacer au-delà des limites définies (*maxBounds*, *maxBoundsViscosity*)

La carte est créée avec la fonction *L.map()*. La méthode *.setView()* est utilisée pour centrer la vue initiale avec un zoom de départ défini.

Deux fichiers de données géographiques sont ensuite préparées : *pathsLayer* (groupe de couches vide prêt à accueillir les chemins) et le path pour les fichiers GeoJSON qui contiendront respectivement les données des chemins des loups et des points d'observation (*cheminsPath* et *geojsonPath*). Enfin un fond de carte est ajouté à l'aide de la fonction *L.tileLayer()*.

```

// #####
// 1. Initialisation de la carte

// Définition du territoire Suisse
const switzerlandBounds = [[45.817993, 5.955911], [47.808455, 10.49205]];

// Initialisation de la carte
const map = L.map("map", {
  maxBounds: switzerlandBounds,
  maxBoundsViscosity: 1.0
}).setView([46.8182, 8.2275], 8);
const pathsLayer = L.layerGroup().addTo(map);
const cheminsPath = './data/chemins.geojson';
const geojsonPath = './data/spot_all.geojson';

// Ajouter le layer Tile
L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
  maxZoom: 19,
  minZoom: 8,
  attribution: "© OpenStreetMap contributors"
}).addTo(map);

```

Le deuxième bloc de code s'occupe de la gestion des clusters.

Une icône SVG en forme de loup est définie dans la constante `wolfIconHtmlCluster`. Un groupe de clusters est créé en utilisant `L.markerClusterGroup()`. La fonction `iconCreateFunction` est utilisée pour personnaliser l'apparence de chaque cluster, avec le nombre de marqueurs et l'icône.

```
//
#####
#####
#####2. Clusters
// Icône SVG (Loup gris)
const wolfIconHtmlCluster = `

<svg height="80px" width="65px" version="1.1" id="_x34_" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
viewBox="0 0 512 512" xml:space="preserve">
<g>
    <path style="fill:#C0B495;" d="M343.736,404.417c0,31.472-25.509,56.984-56.981,56.984h-61.511
    c-31.465,0-56.981-25.512-56.981-56.984l0,c0-31.472,25.516-56.984,56.981-56.984h61.511
    C318.227,347.433,343.736,372.945,343.736,404.417l343.736,404.417z"/>
    <path style="fill:#92959D;" d="M300.341,72.46c0,0,43.416-54.272,54.276-65.126C365.47-3.52,384.46-3.52,389.89,15.472
c5.423,18.996,37.992,116.687,37.992,116.687l512,320.295h-45.516v45.227l-38.898-8.142l-13.566,38.895l-36.179-1
6.281

l-18.09,34.374l-58.898-33.924l-47.834,23.07h5.963l-47.835-23.07l-58.898,33.924l-18.09-34.374l-36.18,16.281l84
.414,357.38

l-38.898,8.142v-45.227H0184.118-188.136c0,0,32.568-97.691,37.992-116.687c5.43-18.992,24.426-18.992,35.28-8.13
8
        c10.853,10.853,54.269,65.126,54.269,65.126h300.341z"/>
    <g>
    <g>
        <polygon style="fill:#FFFFFF;" points="182.728,100.5 147.455,54.37 133.89,100.5 "/>
    </g>
    <g>
        <polygon style="fill:#FFFFFF;" points="329.272,100.5 364.544,54.37 378.11,100.5 "/>
    </g>
    </g>
        <path style="fill:#FFFFFF;" d="M256,403.514h32.562c0,0,27.328,1.084,48.307-19.898c25.326-25.326,35.135-78.019,35.135-78.019
c22.384-3.391,64.322-66.567,18.313-108.314c-48.839-44.321-127.534,3.165-118.486,79.145v74.584l14.38,39.84l256
,394.469v1.806

l-30.217-3.615l14.386-39.84v-74.585c9.041-75.983-69.647-123.469-118.492-79.145c-46.002,41.747-4.071,104.924,1
8.313,108.315
        c0,0,9.816,52.69,35.135,78.015c20.985,20.985,48.307,19.899,48.307,19.899h256v403.514z"/>
    <g>
    <g>
        <path style="fill:#564236;" d="M180.614,252.88h-18.609l6.021,8.162c-0.781,1.733-1.228,3.628-1.228,5.636
c0,7.63,6.178,13.819,13.815,13.819c7.617,0,13.802-6.189,13.802-13.819c194.416,259.062,188.231,252.88,180.614,
252.88z"/>
        </g>
        <g>
            <path style="fill:#564236;" d="M331.386,252.88h18.609l-6.014,8.162c0.775,1.733,1.221,3.628,1.221,5.636
d="M331.386,252.88h18.609l-6.014,8.162c0.775,1.733,1.221,3.628,1.221,5.636`
```

```

c0,7.63-6.179,13.819-13.815,13.819c-7.617,0-13.802-6.189-13.802-13.819C317.584,259.062,323.769,252.88,331.386
,252.88z"/>
    </g>
    </g>
        <path style="fill:#71605B;" d="M283.046,383.839c-5.759,0-27.046,0-27.046,0s-21.294,0-27.046,0
c-16.113,0-26.461,19.56-13.802,35.673c15.779,20.092,30.487,20.713,40.848,20.713c10.354,0,25.063-0.621,40.848-
20.713
            C309.514,403.399,299.159,383.839,283.046,383.839z"/>
    </g>
        <path style="opacity:0.23;fill:#71605B;" d="M427.882,132.159c0,0-32.568-97.691-37.992-116.687
C384.46-3.52,365.47-3.52,354.617,7.334c-10.86,10.853-54.276,65.126-54.276,65.126h-49.659v388.941h36.074
c31.228,0,56.561-25.135,56.948-56.275l16.048,9.242l18.09-34.374l36.179,16.281l13.566-38.895l38.898,8.142v-45.
227H512
            L427.882,132.159z"/>
    </g>
</svg>`

// Configuration clusters
const markers = L.markerClusterGroup({
  iconCreateFunction: (cluster) => {
    // Nb. marqueurs par cluster
    const count = cluster.getChildCount();

    // HTML icône cluster
    return L.divIcon({
      html: `

        <div style="position: relative; text-align: center; width: 60px; height: 60px;">
          <div style="position: absolute; top: 10px; left: 10px; width: 40px; height: 40px;">
            ${wolfIconHtmlCluster}
          </div>
          <div style="position: absolute; top: 0; left: 0; width: 60px; height: 60px; display:
flex; align-items: center; justify-content: center;">
            <div style="background: white; border: 2px solid black; border-radius: 50%; width:
30px; height: 30px; display: flex; align-items: center; justify-content: center; font-size: 14px; font-
weight: bold;">
              ${count} <!-- Affiche le nombre de marqueurs dans le cluster -->
            </div>
          </div>
        </div>`,
      className: '',
      iconSize: [60, 60],
    });
  }
});
map.addLayer(markers);

```

Dans la troisième section nous avons défini les principales variables et mis en place le chargement des fichiers GeoJSON. Ces données alimenteront les différentes fonctionnalités interactives de l'application.

Tout d'abord, plusieurs variables sont initialisées: *pathsByWolf* est utilisée pour stocker les déplacements des loups , *data* contient l'ensemble des données d'observation, *filteredData* sera utilisée pour gérer les données après l'application des filtres. La variable *animatedLayer* est créée pour l'animation des chemins.

Les données sont chargées via des requêtes *fetch*. Chaque entrée inclut des attributs spécifiques comme les coordonnées, l'ID, l'année de l'observation, le canton, le genre. Ces données sont ensuite utilisées pour créer des graphiques (*createYearChart*, *createTimeTravelScatterPlot* et *createCantonChartStatic*). Les points d'observation sont affichés sur la carte à l'aide de la fonction *addAllMarkers*.

Le deuxième fichier, spécifié par *cheminsPath*, contient les données des déplacements (chaque déplacement est associé à l'ID)

```
//
#####
#####3. Données et variables

// Variables
let pathsByWolf = {}; // Chemin loups
let data = []; // Données observation
let filteredData = [] // Données filtré
let animatedLayer = null; // Animation chemins

// Graphiques de defaut
fetch(jsonPath)
  .then(response => response.json())
  .then(geojsonData => {

    // Conversion geojson
    data = geojsonData.features.map(feature => ({
      lat: feature.geometry.coordinates[1],
      lon: feature.geometry.coordinates[0],
      wolfID: feature.properties.joint_wolf_id,
      year: feature.properties.joint_date.split("-")[0],
      canton: feature.properties.joint_nom_canton,
      gender: feature.properties.joint_sexe,
      commune: feature.properties.name,
      date: feature.properties.joint_date
    }));
    // Afficher les graphiques par défaut
    createYearChart(data);
    createCantonChartStatic(data);

    // Afficher les marqueurs par defaut
    addAllMarkers(data);

    // Données pour scatterplot
    const scatterPlotData = geojsonData.features.map(feature => ({
      id: feature.properties.joint_wolf_id,
      longitude: feature.geometry.coordinates[0],
      latitude: feature.geometry.coordinates[1],
      year: new Date(feature.properties.joint_date).getFullYear(),
      gender: feature.properties.joint_sexe
    }));
  });
}
```

```

    // Création scatterplot
    createTimeTravelScatterPlot(scatterPlotData);
}

// Données des déplacements
fetch(cheminsPath)
  .then(response => response.json())
  .then(cheminsData => {
    console.log("Données des déplacements chargées :", cheminsData);

    // Liaison avec ID
    cheminsData.features.forEach(feature => {
      const wolfID = feature.properties.joint_wolf_id;
      if (!pathsByWolf[wolfID]) {
        pathsByWolf[wolfID] = [];
      }
      pathsByWolf[wolfID].push(feature);
    });
  });
}

```

Dans la section suivante nous implémentons les fonctionnalités d'animation et d'interactivité.

La fonction *showWolfPaths(wolfID)* est en charge d'afficher et d'animer les trajets d'un loup spécifique, identifié par son *wolfID*. Les données de trajets sont triées en ordre chronologique pour garantir que les déplacements sont visualisés dans le bon ordre. Avant d'ajouter de nouveaux trajets, toutes les couches précédentes liées aux chemins sont effacées à l'aide de *pathsLayer.clearLayers()*. La fonction *animatePath* est appelée pour visualiser le déplacement progressif du loup sur le chemin, et *animatePath* gère l'animation des déplacements du loup.

Un marqueur est créé à l'aide d'une icône SVG personnalisée représentant un loup. Un intervalle (*setInterval*) met à jour régulièrement la position du marqueur: une fois arrivé à la fin l'animation recommence depuis le début. La vitesse est définie par l'intervalle de temps entre chaque mise à jour (800 ms ici).

```

//#####
##4. Fonctions d'animations, interactivité

//4.1 Animations pour les chemins
###
// Fonction animation chemins
function showWolfPaths(wolfID) {
  const paths = pathsByWolf[wolfID];

  if (!paths || paths.length === 0) {
    console.warn(`Nessun percorso trovato per ${wolfID}`);
    pathsLayer.clearLayers();
    return;
  }

  // Ordre chrono
  const sortedPaths = paths.sort((a, b) => {
    const dateA = new Date(a.properties.joint_date);

```

```

    const dateB = new Date(b.properties.joint_date);
    return dateA - dateB;
});

// Supprimer chemins existants
pathsLayer.clearLayers();

// Lat-long
const coordinates = [];

// Extraction coordonnées dans l'ordre chrono
sortedPaths.forEach(path => {
    const coords = path.geometry.coordinates;

    if (!coords || coords.length === 0) {
        console.warn("Segmento senza coordinate, saltato.");
        return;
    }

    coords.forEach(coord => {
        const latLng = [coord[1], coord[0]];
        coordinates.push(latLng);

        // Point observation
        L.circleMarker(latLng, {
            radius: 5,
            color: 'red',
            weight: 1,
            fillColor: 'white',
            fillOpacity: 0.7
        })
        .bindPopup(`Osservazione: ${JSON.stringify(path.properties)}`)
        .addTo(pathsLayer);
    });
});

// Trace chemin
const line = L.polyline(coordinates, { color: 'black', weight: 2 }).addTo(pathsLayer);

// Zoom auto sur le chemin
const bounds = line.getBounds();
map.fitBounds(bounds, { padding: [20, 20] });

// Démarrer animation
animatePath(coordinates);
}

function animatePath(coordinates) {
    let index = 0;

    // Supprimer animations précédentes
    if (animatedLayer) {
        map.removeLayer(animatedLayer);
    }

    // Nouveau layer animation
    animatedLayer = L.layerGroup().addTo(map);

    // Icône SVG (loup gris)
    const wolfIconHtml =
<svg height="40px" width="40px" version="1.1" id="_x34_" xmlns="http://www.w3.org/2000/svg"

```

```

xmlns:xlink="http://www.w3.org/1999/xlink"
    viewBox="0 0 512 512" xml:space="preserve">
    <g>
        <g>
            <path style="fill:#C0B495;" d="M343.736,404.417c0,31.472-25.509,56.984-56.981,56.984h-61.511
c-31.465,0-56.981-25.512-56.981-56.984l0,0c-31.472,25.516-56.984,56.981-56.984h61.511
C318.227,347.433,343.736,372.945,343.736,404.417L343.736,404.417z"/>
            <path style="fill:#92959D;" d="M300.341,72.46c0,0,43.416-54.272,54.276-65.126C365.47-3.52,384.46-3.52,389.89,15.472
c5.423,18.996,37.992,116.687,37.992,116.687L512,320.295h-45.516v45.227l-38.898-8.142l-13.566,38.895l-36.179-1
6.281
l-18.09,34.374l-58.898-33.924l-47.834,23.07h5.963l-47.835-23.07l-58.898,33.924l-18.09-34.374l-36.18,16.281L84
.414,357.38
l-38.898,8.142v-45.227H0184.118-188.136c0,0,32.568-97.691,37.992-116.687c5.43-18.992,24.426-18.992,35.28-8.13
8
            c10.853,10.853,54.269,65.126,54.269,65.126H300.341z"/>
        <g>
            <g>
                <polygon style="fill:#FFFFFF;" points="182.728,100.5 147.455,54.37 133.89,100.5
" />
            </g>
            <g>
                <polygon style="fill:#FFFFFF;" points="329.272,100.5 364.544,54.37 378.11,100.5
" />
            </g>
            <g>
                <path style="fill:#FFFFFF;" d="M256,403.514h32.562c0,0,27.328,1.084,48.307-19.898c25.326-25.326,35.135-78.019,35.135-78.019
c22.384-3.391,64.322-66.567,18.313-108.314c-48.839-44.321-127.534,3.165-118.486,79.145v74.584l14.38,39.84L256
,394.469v1.806
l-30.217-3.615l14.386-39.84v-74.585c9.041-75.983-69.647-123.469-118.492-79.145c-46.002,41.747-4.071,104.924,1
8.313,108.315
            c0,0,9.816,52.69,35.135,78.015c20.985,20.985,48.307,19.899,48.307,19.899H256V403.514z"/>
            <g>
            <g>
                <path style="fill:#564236;" d="M180.614,252.88h-18.609l6.021,8.162c-0.781,1.733-1.228,3.628-1.228,5.636
d="M331.386,252.88h18.609l-6.014,8.162c0.775,1.733,1.221,3.628,1.221,5.636
c0,7.63-6.179,13.819,13.815,13.819c7.617,0,13.802-6.189,13.802-13.819C194.416,259.062,188.231,252.88,180.614,
252.88z"/>
            </g>
            <g>
                <path style="fill:#564236;" d="M283.046,383.839c-5.759,0-27.046,0-27.046,0s-21.294,0-27.046,0
c-16.113,0-26.461,19.56-13.802,35.673c15.779,20.092,30.487,20.713,40.848,20.713c10.354,0,25.063-0.621,40.848-
20.713
            C309.514,403.399,299.159,383.839,283.046,383.839z"/>
            </g>
            <path style="opacity:0.23;fill:#71605B;" d="M427.882,132.159c0,0-32.568-97.691-37.992-116.687
" />
        </g>
    </g>

```

```

C384.46-3.52,365.47-3.52,354.617,7.334c-10.86,10.853-54.276,65.126-54.276,65.126h-49.659v388.941h36.074
c31.228,0,56.561-25.135,56.948-56.275l16.048,9.242l18.09-34.374l36.179,16.281l13.566-38.895l38.898,8.142v-45.
227H512
    L427.882,132.159z"/>
</g>
</svg>`;

// Marqueur avec icone SVG
const animatedMarker = L.divIcon({
  html: wolfIconHtml,
  className: ''
});

// Ajoute marqueur
const marker = L.marker(coordinates[0], { icon: animatedMarker }).addTo(animatedLayer);

// Mise à jour marqueur
function updateMarker() {
  if (index < coordinates.length - 1) {
    index++;
    marker.setLatLng(coordinates[index]);
  } else {
    index = 0; // Riparte dall'inizio
    marker.setLatLng(coordinates[index]);
  }
}

// Vitesse animation
const animationInterval = setInterval(updateMarker, 800);
animatedLayer.animationInterval = animationInterval;
}

```

Dans la section qui suit nous avons implémenté les marqueurs pour visualiser les emplacements des observations sur la carte et les popups interactifs. La fonction `addAllMarkers` est utilisée pour ajouter ces marqueurs à la carte, tandis que `createWolfIcon` génère une icône personnalisée en SVG représentant un loup, offrant une représentation visuelle unique.

Chaque marqueur est associé à un popup interactif contenant des informations détaillées sur l'observation, telles que la localisation, l'ID du loup, ou la date. Ces marqueurs sont regroupés dans une couche *Leaflet* (*markers*), qui est ensuite ajoutée à la carte principale. Avant d'ajouter de nouveaux marqueurs, tous les marqueurs existants sont supprimés afin d'éviter les duplications, garantissant ainsi un affichage clair et précis.

```

//4.2 Marquers
#####
#####

// Chargement des données chemin
fetch(geojsonPath)
  .then(response => response.json())
  .then(geojsonData => {
    console.log("Dati GeoJSON caricati:", geojsonData);

    // Conversion GeoJSON
    data = geojsonData.features.map(feature => ({
      lat: feature.geometry.coordinates[1],
      lon: feature.geometry.coordinates[0],
      wolfID: feature.properties.joint_wolf_id,
      year: feature.properties.joint_date.split("-")[0],
      canton: feature.properties.joint_nom_canton,
      gender: feature.properties.joint_sexe,
      commune: feature.properties.name,
      date: feature.properties.joint_date
    }));
    console.log("Dati trasformati per i marker:", data);

    // Montrer tous les marquers par défaut
    addAllMarkers(data);
  })
  .catch(error => console.error("Errore nel caricamento dei marker:", error));

// Icône SVG (loup gris)
const wolfIconHtml =
  `<svg height="40px" width="40px" version="1.1" id="_x34_" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  viewBox="0 0 512 512" xml:space="preserve">
    <g>
      <g>
        <path style="fill:#C0B495;" d="M343.736,404.417c0,31.472-25.509,56.984-56.981,56.984h-61.511
        c-31.465,0-56.981-25.512-56.981-56.984l0,0c0-31.472,25.516-56.984,56.981-56.984h61.511
        C318.227,347.433,343.736,372.945,343.736,404.417l343.736,404.417z"/>
        <path style="fill:#92959D;" d="M300.341,72.46c0,0,43.416-54.272,54.276-65.126c365.47-3.52,384.46-3.52,389.89,15.472
        c5.423,18.996,37.992,116.687,37.992,116.687l512,320.295h-45.516v45.227l-38.898-8.142l-13.566,38.895l-36.179-1
        6.281

l-18.09,34.374l-58.898-33.924l-47.834,23.07h5.963l-47.835-23.07l-58.898,33.924l-18.09-34.374l-36.18,16.281l84
.414,357.38

l-38.898,8.142v-45.227H0l84.118-188.136c0,0,32.568-97.691,37.992-116.687c5.43-18.992,24.426-18.992,35.28-8.13
`
```

```

8
      c10.853,10.853,54.269,65.126,54.269,65.126H300.341z" />
    <g>
    <g>
      <polygon style="fill:#FFFFFF;" points="182.728,100.5 147.455,54.37 133.89,100.5 " />
    </g>
    <g>
      <polygon style="fill:#FFFFFF;" points="329.272,100.5 364.544,54.37 378.11,100.5 " />
    </g>
    </g>
      <path style="fill:#FFFFFF;" d="M256,403.514h32.562c0,0,27.328,1.084,48.307-19.898c25.326-25.326,35.135-78.019,35.135-78.019
c22.384-3.391,64.322-66.567,18.313-108.314c-48.839-44.321-127.534,3.165-118.486,79.145v74.584l14.38,39.84L256
,394.469v1.806

1-30.217-3.615l14.386-39.84v-74.585c9.041-75.983-69.647-123.469-118.492-79.145c-46.002,41.747-4.071,104.924,1
8.313,108.315
      c0,0,9.816,52.69,35.135,78.015c20.985,20.985,48.307,19.899,48.307,19.899H256V403.514z" />
    <g>
    <g>
      <path style="fill:#564236;" d="M180.614,252.88h-18.60916.021,8.162c-0.781,1.733-1.228,3.628-1.228,5.636
c0,7.63,6.178,13.819,13.815,13.819c7.617,0,13.802-6.189,13.802-13.819c194.416,259.062,188.231,252.88,180.614,
252.88z" />
      </g>
      <g>
        <path style="fill:#564236;" d="M331.386,252.88h18.6091-6.014,8.162c0.775,1.733,1.221,3.628,1.221,5.636
c0,7.63-6.179,13.819-13.815,13.819c-7.617,0-13.802-6.189-13.802-13.819c317.584,259.062,323.769,252.88,331.386
,252.88z" />
        </g>
        </g>
          <path style="fill:#71605B;" d="M283.046,383.839c-5.759,0-27.046,0-27.046,0s-21.294,0-27.046,0
c-16.113,0-26.461,19.56-13.802,35.673c15.779,20.092,30.487,20.713,40.848,20.713c10.354,0,25.063-0.621,40.848-
20.713
            C309.514,403.399,299.159,383.839,283.046,383.839z" />
          </g>
          <path style="opacity:0.23;fill:#71605B;" d="M427.882,132.159c0,0-32.568-97.691-37.992-116.687
C384.46-3.52,365.47-3.52,354.617,7.334c-10.86,10.853-54.276,65.126-54.276,65.126h-49.659v388.941h36.074
c31.228,0,56.561-25.135,56.948-56.275l16.048,9.242l18.09-34.374l36.179,16.281l13.566-38.895l38.898,8.142v-45.
227H512
            L427.882,132.159z" />
          </g>
        </svg>`;

// Création icône
function createWolfIcon() {
  return L.divIcon({
    className: '',
    html: wolfIconHtml,
    iconSize: [30, 30],
    iconAnchor: [15, 15]
  });
}

```

```

// Ajoute marqueurs
function addAllMarkers(markerData) {
    markers.clearLayers();
    markerData.forEach(item => {
        const wolfData = markerData.filter(data => data.wolfID === item.wolfID);
        const uniqueCommunes = new Set(wolfData.map(data => data.commune));
        const totalDistance = calculateTotalDistance(wolfData);

        const marker = L.marker([item.lat, item.lon], {
            icon: createWolfIcon()
        }).bindPopup(`

            <b>Commune:</b> ${item.commune}<br>
            <b>Idéntifiant:</b> ${item.wolfID}<br>
            <b>Date:</b> ${item.date}<br>
            <b>Distance parcourue à vol d'oiseau:</b> ${totalDistance} km<br>
            <b>Nb. de municipalités avec observations:</b> ${uniqueCommunes.size}
        `);
        markers.addLayer(marker);
    });
    map.addLayer(markers);
}

```

L'objectif du prochain bloc de code est d'afficher des informations détaillées sur les déplacements d'un loup telles que la distance totale parcourue et le nombre de communes visitées dans un infobox.

La fonction calculateTotalDistance calcule la distance totale parcourue par un loup en utilisant les coordonnées de ses différentes observations. En parcourant les données des déplacements d'un loup, elle calcule la distance entre chaque paire de points consécutifs à l'aide de la bibliothèque turf.js. La fonction updateWolfAnalysisPanel est utilisée pour afficher les analyses spécifiques d'un loup dans une infobox dédiée, identifiée par l'élément HTML wolf-analysis-panel. Cette fonction accepte trois paramètres: ID, distance et communes.

```

//4.3 Infobox
#####
#####
#####

// Calcul distance total
function calculateTotalDistance(wolfData) {
    let totalDistance = 0;
    for (let i = 1; i < wolfData.length; i++) {
        const prev = wolfData[i - 1];
        const curr = wolfData[i];
        const distance = turf.distance([prev.lon, prev.lat], [curr.lon, curr.lat], { units: 'kilometers' });
        totalDistance += distance;
    }
    return totalDistance.toFixed(2); // 2 decimal
}

// Mise à jour infobox
function updateWolfAnalysisPanel(wolfID, totalDistance, uniqueCommunes) {
    const panel = document.getElementById("wolf-analysis-panel");
    panel.innerHTML = `

        <h3>Analisi del Lupo: ${wolfID}</h3>
        <p><b>Distanza totale percorsa:</b> ${totalDistance} km</p>
        <p><b>Numero di comuni visitati:</b> ${uniqueCommunes}</p>
    `;
}

```

La partie du code suivante implémente deux fonctions principales, *applyFilters* et *resetFilters*. Ces fonctions permettent de personnaliser l'affichage des marqueurs, des chemins et des graphiques en fonction des critères choisis.

La fonction *applyFilters* applique les filtres sélectionnés pour afficher uniquement les données correspondant aux critères spécifiés. Les valeurs des filtres sont extraites des éléments HTML correspondants et puis stockées dans la variable *filteredData*. Si une animation est en cours, elle est arrêtée et supprimée, même chose avec les chemins. La fonction *updateCharts* est appelée pour adapter les graphiques aux données filtrées, et *resetFilters* permet de réinitialiser tous les filtres et de revenir à l'affichage par défaut. Le zoom de la carte est restauré pour inclure l'ensemble de la région d'étude.

```
//4.4 Filtres
#####
#####

// Fonction application filtres
function applyFilters() {
    const gender = document.getElementById("filter-gender").value;
    const year = document.getElementById("filter-year").value;
    const canton = document.getElementById("filter-canton").value;
    const wolfID = document.getElementById("filter-wolf-id").value;

    // Filtrer les données
    filteredData = data.filter(item => {
        return (!gender || item.gender === gender) &&
            (!year || item.year === year) &&
            (!canton || item.canton === canton) &&
            (!wolfID || item.wolfID === wolfID);
    });

    // Supprimer animations précédentes
    if (animatedLayer) {
        map.removeLayer(animatedLayer);
        animatedLayer = null;
    }

    // Supprimer marqueurs si un loup est sélectionné
    if (wolfID) {
        markers.clearLayers();
        showWolfPaths(wolfID);
    } else {
        // Si aucun ID sélectionné afficher les marqueurs filtrés
        addAllMarkers(filteredData);
        pathsLayer.clearLayers();
    }

    // Mise à jour des graphiques avec données filtrées
    updateCharts(filteredData);
}

function resetFilters() {
    // Réinitialiser valeurs filtre
    document.getElementById("filter-gender").value = "";
    document.getElementById("filter-year").value = "";
    document.getElementById("filter-canton").value = "";
    document.getElementById("filter-wolf-id").value = "";
```

```

// Restaurer données d'origine
filteredData = data;

// Mise à jour carte
markers.clearLayers();
addAllMarkers(filteredData);

// Supprimer les chemins existants
pathsLayer.clearLayers();

// Arrêter l'animation active
if (animatedLayer) {
    map.removeLayer(animatedLayer);
    clearInterval(animatedLayer.animationInterval);
    animatedLayer = null;
}

// Rétablissement zoom min
map.fitBounds(switzerlandBounds, { padding: [20, 20] });

// Mise à jour graphiques
updateCharts(filteredData);
}

```

Dans la prochaine section du javascript, nous avons développé des fonctionnalités pour afficher et mettre à jour les graphiques de manière dynamique en fonction des données filtrées et nous avons créé le graphique à barres des observation par année.

La fonction *updateCharts* est responsable de mettre à jour tous les graphiques lorsqu'un filtrage est appliqué. Pour le graphique des observations par année, un conteneur SVG est créé à l'intérieur de l'élément HTML désigné. La taille du SVG s'adapte dynamiquement en fonction des dimensions du conteneur afin de garantir une visualisation réactive. Les données sont regroupées par année à l'aide de la fonction *d3.rollups*, qui compte le nombre d'observations pour chaque année et ensuite triées chronologiquement.

L'échelle X utilise *d3.scaleBand* pour représenter les années avec un espacement uniforme et l'échelle Y utilise *d3.scaleLinear* pour représenter le nombre d'observations avec un domaine allant de 0 au maximum. Les étiquettes de l'axe sont affichées avec une rotation de -45° pour optimiser la lisibilité. L'axe Y est implicite car les valeurs sont représentées directement sur les barres.

Un écouteur d'événement est ajouté avec *window.addEventListener("resize")* pour s'assurer que les graphiques sont redimensionnés lorsque la taille de la fenêtre change.

```

// 4.5 Graphiques
#####
function updateCharts(filteredData) {
    // Mise à jour des graphiques
    createYearChart(filteredData);
    createCantonChartStatic(filteredData);
    createTimeTravelScatterPlot(originalData);
}

// 4.5.a Observation par année
function createYearChart(data) {
    const container = document.getElementById("year-chart");

```

```

// Dimensions et marges dynamique
const margin = { top: 40, right: 10, bottom: 40, left: 10 };
const width = container.clientWidth - margin.left - margin.right;
const height = container.clientHeight - margin.top - margin.bottom;

// Reactivité SVG
const svg = d3.select("#year-chart").html("").append("svg")
  .attr("viewBox", `0 0 ${width + margin.left + margin.right} ${height + margin.top + margin.bottom}`)
  .attr("preserveAspectRatio", "xMidYMid meet")
  .append("g")
  .attr("transform", `translate(${margin.left},${margin.top})`);

// Préparation données
const groupedData = d3.rollups(data, v => v.length, d => +d.year).sort((a, b) => a[0] - b[0]);

// X
const x = d3.scaleBand()
  .domain(groupedData.map(d => d[0]))
  .range([0, width])
  .padding(0.1);

// Y
const y = d3.scaleLinear()
  .domain([0, d3.max(groupedData, d => d[1])])
  .range([height, 0]);

// Axe X
svg.append("g")
  .attr("transform", `translate(0, ${height})`)
  .call(d3.axisBottom(x).ticks(Math.floor(width / 50)))
  .selectAll("text")
  .style("font-family", "Trebuchet MS")
  .style("font-size", `${Math.max(width / 50, 8)}px`)
  .attr("transform", "rotate(-45)")
  .style("text-anchor", "end");

// Barres
svg.selectAll(".bar")
  .data(groupedData)
  .enter().append("rect")
  .attr("class", "bar")
  .attr("x", d => x(d[0]))
  .attr("y", d => y(d[1]))
  .attr("width", x.bandwidth())
  .attr("height", d => height - y(d[1]))
  .attr("fill", "#D9D9D9");
svg.selectAll(".bar-label")
  .data(groupedData)
  .enter().append("text")
  .attr("class", "bar-label")
  .attr("x", d => x(d[0]) + x.bandwidth() / 2)
  .attr("y", d => y(d[1]) - 5)
  .attr("text-anchor", "middle")
  .style("font-family", "Trebuchet MS")
  .style("font-size", `${Math.max(width / 50, 8)}px`)
  .style("font-weight", "bold")
  .text(d => d[1]);

// Titre

```

```

    svg.append("text")
      .attr("x", width / 2)
      .attr("y", -margin.top / 2)
      .attr("text-anchor", "middle")
      .style("font-family", "Trebuchet MS")
      .style("font-size", `${Math.max(width / 25, 14)}px`)
      .style("font-weight", "bold")
      .text("Répartition des observations par année");
}

// Mise à jour avec redimensionnement
window.addEventListener("resize", () => {
  createYearChart(data);
  createCantonChartStatic(data);
});

```

Dans cette section, nous mettons en place le graphique représentant la répartition des observations par canton.

La fonction `createCantonChartStatic` génère un graphique en barres, et un conteneur SVG réactif est initialisé dans l'élément HTML correspondant. Les dimensions du graphique sont ajustées dynamiquement en fonction de la taille du conteneur.

Les données sont regroupées par canton à l'aide de la fonction `d3.rollups`, qui compte le nombre d'observations pour chaque canton. Les cantons sont ensuite triés par nombre décroissant. L'échelle X utilise `d3.scaleBand` pour attribuer un espace équitable à chaque canton et l'échelle Y utilise `d3.scaleLinear` pour représenter le nombre d'observations avec une hauteur proportionnelle. L'axe X affiche les noms des cantons avec une rotation de -45° pour une meilleure lisibilité et l'axe Y est encore une fois implicite. Un écouteur d'événements `resize` est implémenté pour redimensionner automatiquement le graphique lorsque la fenêtre du navigateur est redimensionnée.

```

// 4.5.b Observation par canton
function createCantonChartStatic(data) {
  const container = document.getElementById("canton-chart");

  // Dimension et marges dynamiques
  const margin = { top: 40, right: 10, bottom: 40, left: 10 };
  const width = container.clientWidth - margin.left - margin.right;
  const height = container.clientHeight - margin.top - margin.bottom;

  // Reactivité SVG
  const svg = d3.select("#canton-chart").html("").append("svg")
    .attr("viewBox", `0 0 ${width + margin.left + margin.right} ${height + margin.top + margin.bottom}`)
    .attr("preserveAspectRatio", "xMidYMid meet")
    .append("g")
    .attr("transform", `translate(${margin.left}, ${margin.top})`);

  const cantonCounts = d3.rollups(data, v => v.length, d => d.canton).sort((a, b) => b[1] - a[1]);

  // X
  const x = d3.scaleBand()
    .domain(cantonCounts.map(d => d[0]))
    .range([0, width])
    .padding(0.1);

  // Y
  const y = d3.scaleLinear()
    .domain([0, d3.max(cantonCounts, d => d[1])]);

```

```

    .range([height, 0]);

// Axe X
svg.append("g")
    .attr("transform", `translate(0, ${height})`)
    .call(d3.axisBottom(x).ticks(Math.floor(width / 50)))
    .selectAll("text")
    .style("font-family", "Trebuchet MS")
    .style("font-size", `${Math.max(width / 50, 8)}px`)
    .attr("transform", "rotate(-45)")
    .style("text-anchor", "end");

// Barres
svg.selectAll(".bar")
    .data(cantonCounts)
    .enter().append("rect")
    .attr("class", "bar")
    .attr("x", d => x(d[0]))
    .attr("y", d => y(d[1]))
    .attr("width", x.bandwidth())
    .attr("height", d => height - y(d[1]))
    .attr("fill", "#D9D9D9");
svg.selectAll(".bar-label")
    .data(cantonCounts)
    .enter().append("text")
    .attr("class", "bar-label")
    .attr("x", d => x(d[0]) + x.bandwidth() / 2)
    .attr("y", d => y(d[1]) - 5)
    .attr("text-anchor", "middle")
    .style("font-family", "Trebuchet MS")
    .style("font-size", `${Math.max(width / 50, 8)}px`)
    .style("font-weight", "bold")
    .text(d => d[1]);

// Titre
svg.append("text")
    .attr("x", width / 2)
    .attr("y", -margin.top / 2)
    .attr("text-anchor", "middle")
    .style("font-family", "Trebuchet MS")
    .style("font-size", `${Math.max(width / 25, 14)}px`)
    .style("font-weight", "bold")
    .text("Répartition des observations par Canton");
}

// Redimensionnement
window.addEventListener("resize", () => createCantonChartStatic(data));

```

Dans la dernière section du javascript, nous mettons en place un scatter plot interactif qui visualise les positions des observations de loups (latitude et longitude) en fonction des années sélectionnées.

Un conteneur SVG réactif est initialisé à l'intérieur de l'élément HTML. Les dimensions du graphique sont dynamiques et s'adaptent à la taille du conteneur. Échelles X et Y représentent la longitude et latitude calculées avec `d3.scaleLinear`. Les limites du domaine (`d3.extent`) sont ajustées pour englober toutes les longitudes.

La fonction `update` filtre les données pour une année spécifique, sélectionnée via le time slider. Chaque point est coloré en fonction du genre du loup : bleu pour les mâles et rose pour les femelles. Les nouveaux points sont ajoutés pour chaque année sélectionnée, et les points obsolètes sont supprimés. Un écouteur d'événements `on("input")` est associé au curseur temporel pour détecter les changements sélectionnée.

Un écouteur d'événements `resize` est ajouté pour que le graphique s'adapte automatiquement à la taille de l'écran. En cas de redimensionnement, le graphique est régénéré, et les données pour l'année sélectionnée sont réaffichées.

```
// 4.5.c Graphique lat-long voyage dans le temps
function createTimeTravelScatterPlot(originalData) {
    const container = document.getElementById("scatter-plot");

    // Dimension et marges reactifs
    const margin = { top: 40, right: 20, bottom: 40, left: 50 };
    const width = container.clientWidth - margin.left - margin.right;
    const height = container.clientHeight - margin.top - margin.bottom;

    // Reactivité SVG
    const svg = d3.select("#scatter-plot").html("").append("svg")
        .attr("viewBox", `0 0 ${width + margin.left + margin.right} ${height + margin.top + margin.bottom}`)
        .attr("preserveAspectRatio", "xMidYMid meet")
        .append("g")
        .attr("transform", `translate(${margin.left}, ${margin.top})`);

    // X
    const xScale = d3.scaleLinear()
        .domain(d3.extent(originalData, d => d.longitude))
        .nice()
        .range([0, width]);

    // Y
    const yScale = d3.scaleLinear()
        .domain(d3.extent(originalData, d => d.latitude))
        .nice()
        .range([height, 0]);

    // Axe X
    svg.append("g")
        .attr("transform", `translate(0, ${height})`)
        .call(d3.axisBottom(xScale).ticks(Math.floor(width / 50)))
        .selectAll("text")
        .style("font-family", "Trebuchet MS")
        .style("font-size", `${Math.max(width / 50, 8)}px`)
        .style("font-weight", "bold");

    // Axe Y
    svg.append("g")
        .call(d3.axisLeft(yScale).ticks(Math.floor(height / 50)))
        .selectAll("text")
        .style("font-family", "Trebuchet MS")
        .style("font-size", `${Math.max(height / 50, 8)}px`)
        .style("font-weight", "bold");
}
```

```

// Selection selon année
function update(year) {
    const filteredData = originalData.filter(d => d.year === year);
    const points = svg.selectAll("circle")
        .data(filteredData, d => d.id);

    points.enter()
        .append("circle")
        .attr("cx", d => xScale(d.longitude))
        .attr("cy", d => yScale(d.latitude))
        .attr("r", Math.max(width / 100, 3))
        .attr("fill", d => d.gender === "M" ? "blue" : "pink")
        .merge(points)
        .transition()
        .duration(500)
        .attr("cx", d => xScale(d.longitude))
        .attr("cy", d => yScale(d.latitude));

    points.exit().remove();
}

// Titre
svg.append("text")
    .attr("x", width / 2)
    .attr("y", -margin.top / 2)
    .attr("text-anchor", "middle")
    .style("font-family", "Trebuchet MS")
    .style("font-size", `${Math.max(width / 25, 14)}px`)
    .style("font-weight", "bold")
    .text("Voyage dans le temps");

// Etiquette X
svg.append("text")
    .attr("x", width / 2)
    .attr("y", height + margin.bottom - 9)
    .attr("text-anchor", "middle")
    .text("Longitude")
    .style("font-size", `${Math.max(width / 50, 9)}px`)
    .style("font-family", "Trebuchet MS")
    .style("font-weight", "bold");

// Etiquette Y
svg.append("text")
    .attr("x", -height / 2)
    .attr("y", -margin.left + 16)
    .attr("text-anchor", "middle")
    .attr("transform", "rotate(-90)")
    .text("Latitude")
    .style("font-size", `${Math.max(height / 50, 9)}px`)
    .style("font-family", "Trebuchet MS")
    .style("font-weight", "bold");

// Time-slider
d3.select("#time-slider")
    .on("input", function () {
        const year = +this.value;
        d3.select("#year-label").text(year);
        update(year);
    });
}

```

```

        update(1999);

    // Redimensionnement
    window.addEventListener("resize", () => {
        createTimeTravelScatterPlot(originalData);
        const year = +d3.select("#time-slider").node().value;
        update(year);
    });
}

```

3.4 CSS

Le fichier CSS définit le style global de la page web en organisant les différents éléments. La mise en page repose sur un conteneur principal configuré en flexbox, garantissant une structure responsive qui s'adapte à la taille de l'écran.

L'en-tête est positionné en haut, tandis que la carte occupe une large zone centrale avec les graphiques positionnés verticalement sur le côté droit. Les graphiques sont uniformisés avec un fond blanc, des bordures et une ombre. Le curseur temporel permet de naviguer le scatter plot est placé en dessous de ce dernier tandis que le conteneur de filtres est positionné en bas à gauche. Enfin, une section en bas à droite affiche les sources de données. Pour des raisons esthétiques on a cherché de faire correspondre les limites extérieures des différents conteneurs

```

/* Layout */
body {
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    height: 100vh;
    width: 100vw;
    overflow: hidden;
}

/* Titre */
#header {
    text-align: center;
    position: absolute;
    z-index: 1000;
    top: 5px;
    left: 10px;
    width: 97%;
    height: 50px;
    padding: 10px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    border: 1px solid black;
    border-radius: 5px;
    background-color: #cecece;
    box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.1);
}

#header h1 {
    margin: 0;
    font-size: 24px;
}

```

```

    color: #333;
    font-family: "Trebuchet MS", sans-serif;
}

#header h2 {
    margin: 5px 0 0;
    font-size: 16px;
    color: #666;
    font-family: "Trebuchet MS", sans-serif;
}

/* Carte */
#map {
    position: absolute;
    top: 80px;
    left: 10px;
    right: 25%;
    bottom: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
}

/* Graphiques */
#year-chart, #canton-chart, #scatter-plot {
    position: absolute;
    background: white;
    border: 1px solid #ddd;
    box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
    border-radius: 5px;
    overflow: hidden;
    z-index: 1002;

    width: 23.5%;
    height: 27%;
}

/* Position */
#canton-chart {
    top: 9.25%;
    right: 0.5%;
}

#year-chart {
    top: 37%;
    right: 0.5%;
}

#scatter-plot {
    top: 65%;
    right: 0.5%;
    height: 29%;
}

#time-slider-container {
    position: absolute;
    bottom: 25%;
    right: 6%;
    height: 10%;
    width: 10%;
    display: flex;
}

```

```

    align-items: center;
    justify-content: center;
    z-index: 1100;
}

#time-slider {
    height: 100%;
    width: 200px;
    z-index: 1110;
}

#year-label {
    font-size: 10px;
    font-weight: bold;
    font-family: "Trebuchet MS", sans-serif;
}

/* Container des filtres */
#filters {
    position: absolute;
    bottom: 15px;
    left: 20px;
    display: flex;
    height: 25px;
    justify-content: space-around;
    align-items: center;
    padding: 10px;
    background-color: #f5f5f5;
    box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.1);
    border-top: 1px solid #ddd;
    border-bottom: 1px solid #ddd;
    z-index: 1000;
}

/* Filtres */
#filters select {
    padding: 5px;
    font-size: 10px;
    margin: 10px;
    font-family: 'Trebuchet MS';
    font-weight: normal;
}

#reset-filters {
    padding: 5px 10px;
    font-size: 10px;
    font-family: 'Trebuchet MS';
    background-color: #e74c3c;
    color: white;
    border: none;
    border-radius: 3px;
    cursor: pointer;
    margin-left: 10px;
}

#reset-filters:hover {
    background-color: #c0392b;
}

#filters label {

```

```
margin-right: 5px;
font-size: 10px;
font-family: 'Trebuchet MS';
font-weight: bold;
}

#data-sources {
position: absolute;
bottom: 1%;
right: 0.5%;
width: 23.5%;
text-align: center;
font-size: 9px;
font-family: "Trebuchet MS", sans-serif;
color: #555;
background-color: #f9f9f9;
border: 1px solid #ddd;
border-radius: 5px;
z-index: 1000;
}
```

4. Résultats

La carte interactive est navigable sur le lien suivant : https://ntrojan.github.io/geovis2_versionfinale/ jusqu'au 26 février.

La carte est présentée comme dans les images à la fin de la section. On s'est efforcé d'utiliser une mise en page qui occupe tout l'espace disponible, en donnant la priorité à la carte mais en conservant aux graphiques une taille qui permette une visualisation claire. Pour améliorer l'esthétique, des marges régulières ont été maintenues entre les différents éléments présents. Comme nous l'avons déjà mentionné, l'application est conçue pour être visualisée entièrement à l'intérieur de l'écran, sans possibilité de défilement. Le contenu s'adapte également aux différentes tailles et formes de la fenêtre de visualisation.

Les filtres mettent à jour à la fois le contenu de la carte et celui des deux premiers graphiques, de sorte que des informations spécifiques à la sélection sont affichées en fonction des critères choisis. Le fait d'appuyer sur la touche *reset* rétablit tous les marqueurs et les graphiques avec toutes les données. Le troisième graphique, le nuage de points de la répartition géographique dans le temps, est itératif grâce à un curseur de temps qui permet de voyager dans le temps pour voir son évolution.

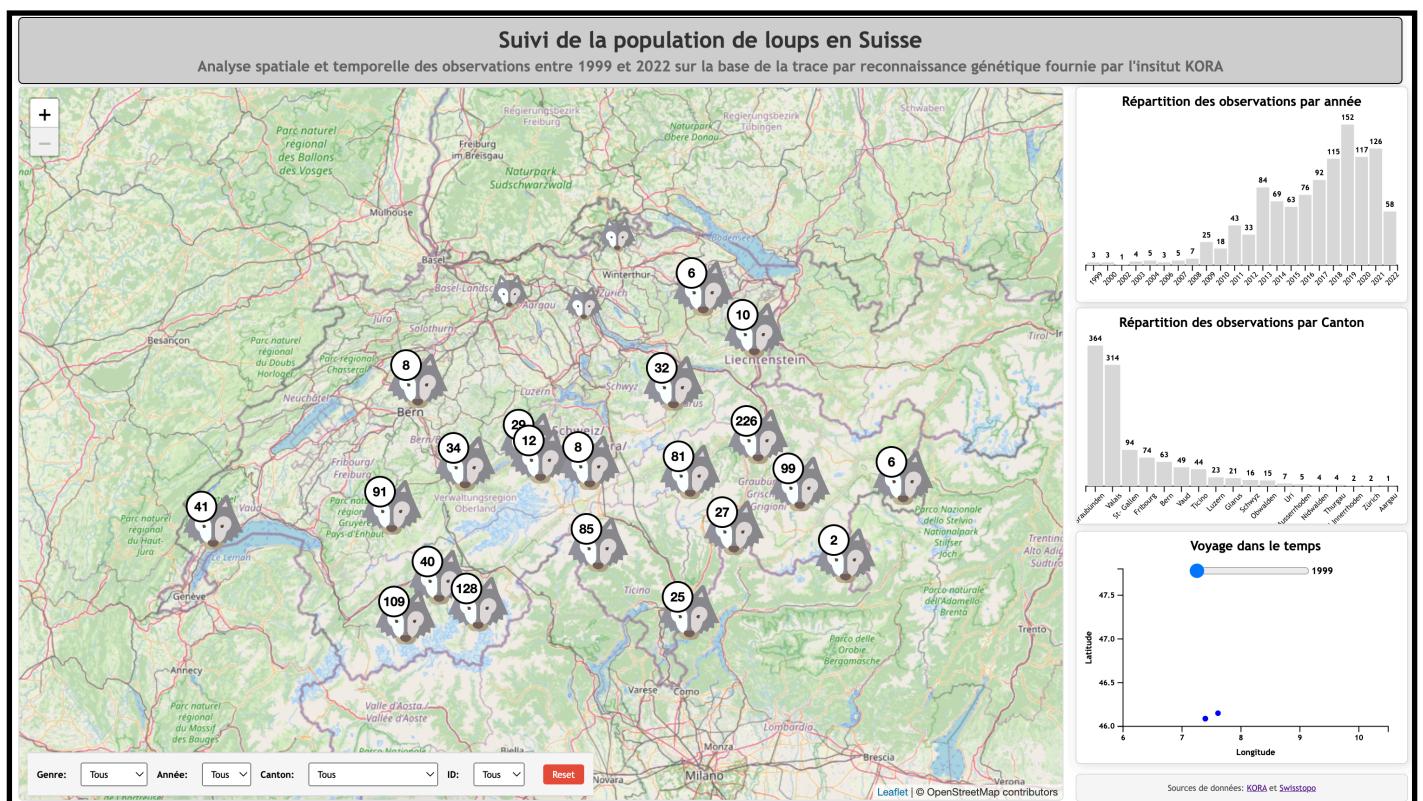


Figure 1: visualisation par défaut

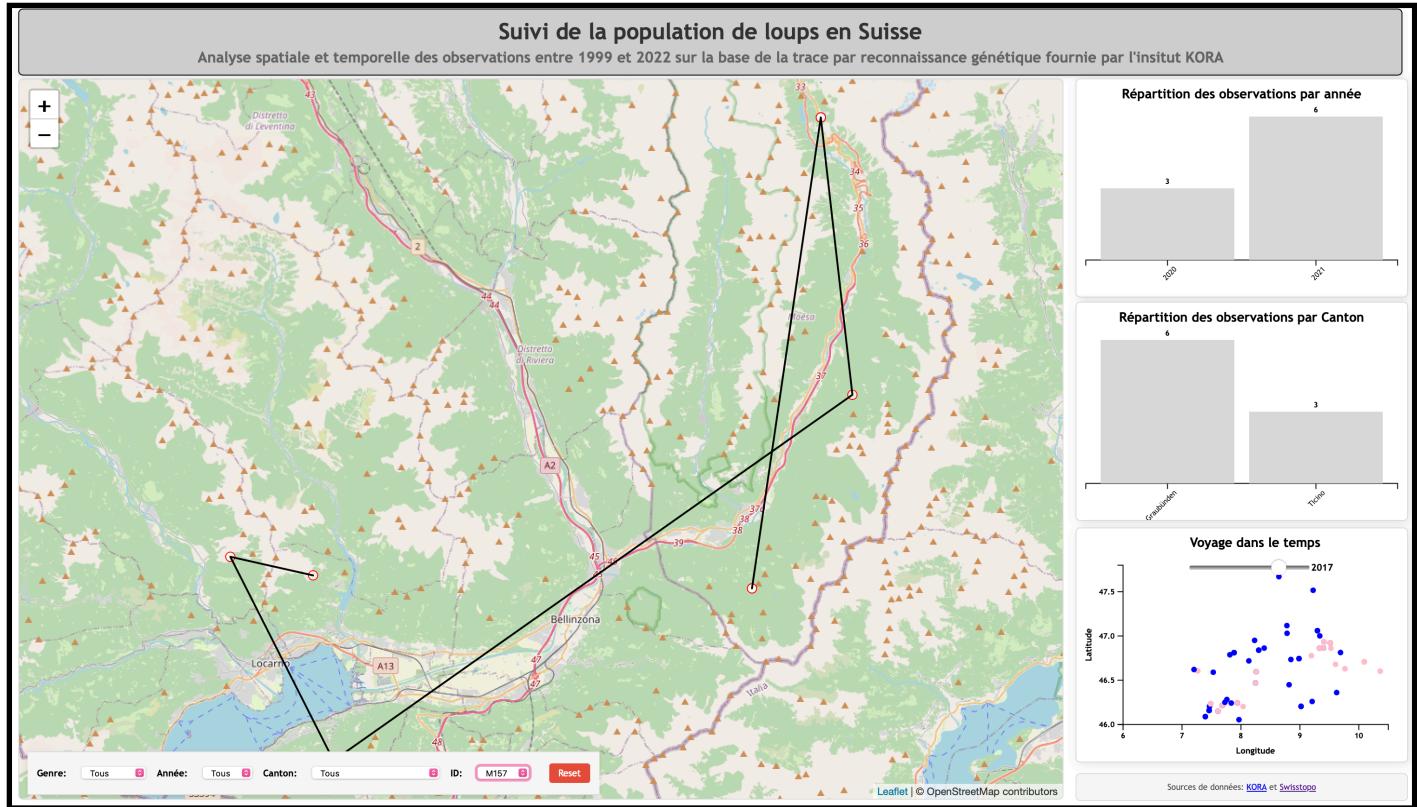


Figure 2: ID sélectionné

5. Limites de l'application

Une première limitation de la carte créée concerne l'approximation de la plupart des observations. Dans le jeu de données source fourni par le KORA, les observations ont été localisées via les hameaux des municipalités si elles ne se trouvaient pas dans la municipalité. Dans le fichier utilisé pour géoréférencer les observations, les hameaux ne sont malheureusement pas disponibles et nous avons donc dû nous rapprocher de la municipalité. Une autre limite à la fiabilité des données concerne les déplacements des individus. Ceux-ci sont calculés comme la distance à vol d'oiseau entre deux observations consécutives et sont donc une approximation basée sur les données disponibles et ne reflètent pas l'itinéraire réel emprunté. Il en va de même pour la distance parcourue calculée.

Une autre limite du travail est la présentation à l'écran. Nous avons d'abord utilisé une présentation « fixe » conçue pour être visualisée à partir de notre ordinateur portable (même modèle). Une fois le projet terminé et l'application testée par des amis, nous nous sommes rendu compte qu'elle n'était pas présentable et avons modifié le fichier css et une partie du javascript en conséquence. La version finale a une mise en page dynamique qui s'adapte à tous les types d'écran mais a quelques limitations. La première est la faible adaptabilité aux petits écrans et aux écrans verticaux. La seconde est le chevauchement de certains conteneurs lorsque la page est trop réduite (figure 3). Par contre le test sur grand écran était bon. Toujours en ce qui concerne la réduction de la page, les titres des graphiques ne suivent pas le comportement dynamique du contenu et des conteneurs et risquent de disparaître en dehors de leurs conteneurs (figure 3). Une dernière limitation de l'application est que la manipulation de la taille de la page web avec l'application ouverte risque de provoquer un plantage et la page doit être rechargée.

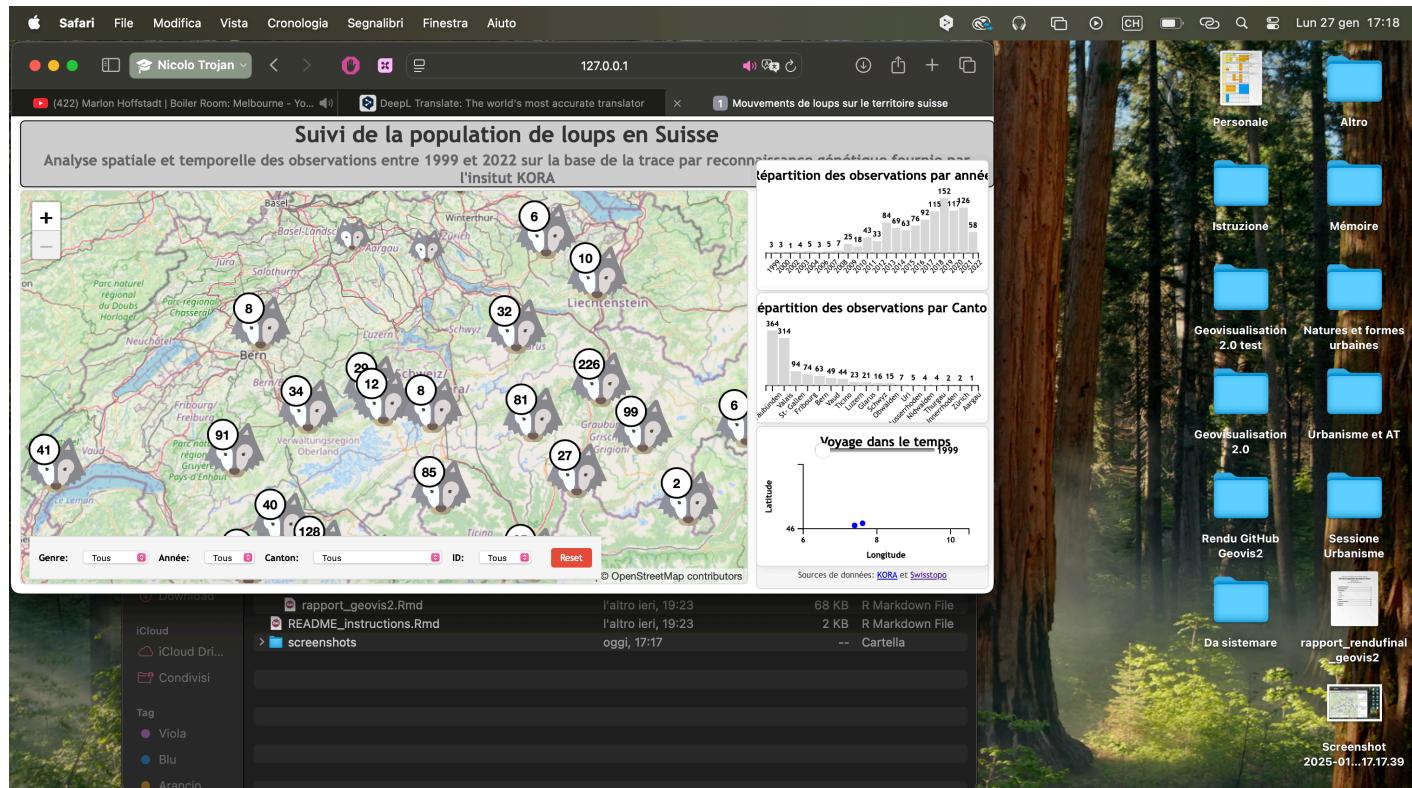


Figure 3: problèmes de mise à l'échelle (chevauchement des éléments et comportement des titres des graphiques)

6. Conclusion

En conclusion, l'ouvrage montre l'évolution de la population de loups en Suisse de manière interactive et claire. La tendance à la hausse de la population et les informations sur les zones les plus touchées sont bien visibles, et d'autres données intéressantes sur les déplacements des individus ou la longévité de leur séjour sur le territoire sont également facilement accessibles à partir des infobox et des graphiques.

Ce rapport peut être utilisé pour reproduire le projet, et cette reproduction ne doit pas nécessairement être liée à la présence du loup ou d'un autre animal sur le territoire, mais peut avoir différentes fonctionnalités pour différents thèmes caractérisés par le déplacement d'éléments dans l'espace.

7. Références

- Office fédéral de l'environnement (OFEV). s. d. « Loup : le Conseil fédéral met en vigueur la régulation préventive des meutes ». Consulté 22 décembre 2024 (<https://www.bafu.admin.ch/bafu/fr/home/themes/biodiversite/communiques.msg-id-98407.html>).
- Pro Natura. s. d. « Histoire du retour du loup en Suisse (1995-2021) | Pro Natura ». Consulté 22 décembre 2024 (<https://www.pronatura.ch/fr/loup>). RSI. 2023. « Aumentano i lupi, ma scendono le predazioni - RSI ». Consulté 22 décembre 2024 (<https://www.rsi.ch/info/svizzera/Aumentano-i-lupi-ma-scendono-le-predazioni--2029759.html>).