
Geoinformatik: Webmapping

Sommersemester 2020

# Projektbericht

Gruppe 2: *Little Green Friends*

Lukas Endres 11849504,

Johanna Roll, 11917354

Elsa Ventruba 1341692

Innsbruck, 17.06.2020

# Inhaltsverzeichnis

# 1. Kurzbeschreibung des Projekts

Zum Abschluss des Methodenkurses Webmapping im Sommersemester 2020 wurde im Rahmen der Lehrveranstaltung eine webbasierte Karte erstellt, die Daten aus dem US-Bundesstaat Connecticut veranschaulicht. Die Daten wurden von der regionalen Regierung erhoben und stellen die Todesfälle aufgrund von Drogenmissbrauch im Zeitraum zwischen 2012 und 2018 dar. Die auf Connecticut Open Data im JSON-Format frei zur Verfügung gestellten Daten werden dabei in einer Karte dargestellt, die auf Basis von Leaflet die einzelnen Todesfälle lokalisiert und mit unterschiedlichen Visualisierungen auf der Karte abbildet. Die Informationen sind in dem Datensatz als Arrays gesammelt, in welchem die Metadaten und Daten zusammengefasst sind.

In der interaktiven Karte können die Todesfälle grundsätzlich auf zwei Arten dargestellt werden: Die einzelnen Fälle mit genaueren Informationen im Popup des Markers, welcher am Ort des Todes platziert ist, und die Gesamtzahl der Fälle pro County, visualisiert mit proportionaler Kreisgröße und Popup. Die einzelnen Fälle können in der Layercontrol nach Geschlecht gefiltert dargestellt werden. Daneben kann eine zeitliche Animation der monatlichen Fälle pro County ausgeführt werden, die über ein Slider-Element unterhalb der Karte gestartet und abgespielt werden kann.

Als informatives Gerüst rund um die Daten dienen zwei weitere Webseiten, die zum einen grundlegende Informationen zu den häufigsten genannten und festgestellten Drogen bereitstellen und zum anderen die Demographie und ökonomische Aufstellung des Bundesstaates kurz beleuchten und mit dem Rest der USA in Vergleich setzen. Über ein dynamisches Span-Element im Header der Webseite kann zwischen den einzelnen Seiten navigiert werden.

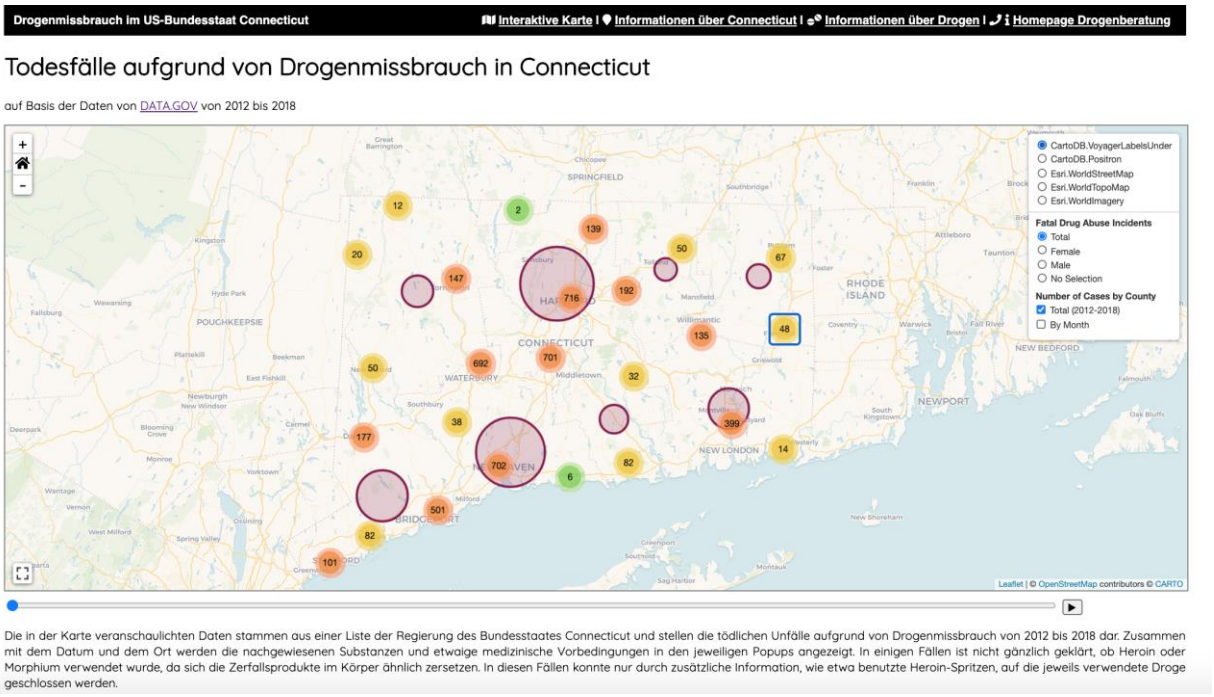


*Abbildung 1: Kartendarstellung (eigene Abbildung) und Seitendarstellung der index.html*

Der JSON-Datensatz, der die Basis für die Karte darstellt, ist in Metadaten und Daten unterteilt, wie in Abbildung 2 und 3 gezeigt wird.

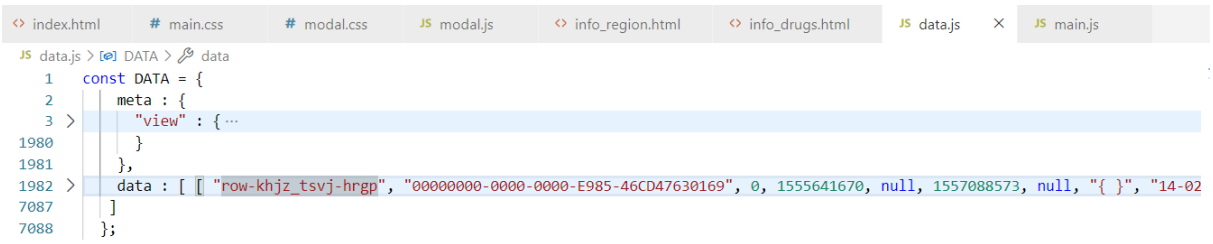


*Abbildung 2: Unterteilung des Datensatzes im JSON-Format*

Innerhalb der Metadaten wurden die einzelnen Indizes der Arrays aus `data` beschrieben. In der interaktiven Karte wurden folgende Informationen verarbeitet:

- Datum
- Geschlecht
- Alter
- Todesort (Ort, Stadt, Staat)
- Todesursache
- medizinische Vorerkrankungen
- nachgewiesene Substanzen


*Abbildung 3: Auszug aus den Meta-Daten*

Die beiden zusätzlichen Webseiten bieten des Weiteren einerseits Informationen zu dem Bundesstaat Connecticut allgemein sowie dem Drogenkonsum in den USA und im Bundesstaat und andererseits zu den in dieser Erhebung an den häufigsten nachgewiesenen Drogen, deren Wirkung und Einteilung. Die Texte sind dabei durch Grafiken und Bilder unterstützt.

Die von dem verwendeten Datensatz ausgehenden Zahlen (z.B. Anzahl der Todesfälle) wurden dabei dynamisch abgerufen und in den Text eingebunden. Um die Übersichtlichkeit der einzelnen .js-Datein zu erhalten wurde die Dynamisierung an info_html.js ausgelagert und über ein `src defer` an die .html-Seite verbunden.

Das Ergebnis ist unter https://littlegreenfriends.github.io/ abrufbar.

# 2. Implementierungsschritte

Nach der eingänglichen Beschreibung des Projektes wird im Folgenden auf die Implementierungsschritte eingegangen und dargestellt, wie das Projekt von seinem Basisgerüst aus stetig angewachsen ist. Zu Anfang wurden, um zu große Ähnlichkeiten zu den in der Lehrveranstaltung behandelten Beispielen zu vermeiden, Daten aus anderen Regionen und Bereichen gesucht, wobei vorrangig auf das Datenformat Wert gelegt wurde. Schlussendlich wurden die von der lokalen Regierung des US-Bundesstaates Connecticut veröffentlichten Daten zu Accidental Drug Related Deaths 2012-2018 verwendet. Das vorliegende Format war im Gegensatz zu den in der Lehrveranstaltung verwendeten Datensätzen kein GeoJSON, sondern ein JSON-File, in dem die Daten wie oben bereits erwähnt in `data` als Arrays festgehalten waren und dementsprechend anders abgerufen werden müssen.

Die Grundstruktur der Webseiten wird von drei HTML-Seiten gebildet:

- index.html https://littlegreenfriends.github.io/index.html
- info_region.html https://littlegreenfriends.github.io/info_region.html
- info_drugs.html https://littlegreenfriends.github.io/info_drugs.html

Basierend auf Leaflet wurde zunächst der Rahmen für das Kartenelement geschaffen und im main.css und main.js die ersten Style-Anpassungen vorgenommen, um zum Beispiel Header und Icons einzubauen und baseMaps einzufügen. Durch die Funktion `drawAccidents` wurden die Daten daraufhin das erste Mal abgerufen und in der Karte veranschaulicht, zuerst noch alle einzeln, was die Rechendauer der Webseite ungemein erhöhte und als Lösung dafür dann mittels dem MarkerCluster Plugin.
In den weiteren Schritten wurde die Darstellung der Daten in der Karte bearbeitet, um die oben genannten Filter und Cluster-Funktionen passend anzuzeigen. Dazu wurden zum Beispiel die Polygone mittel einem Hover angepasst und an die Auswahlmöglichkeiten nach Geschlecht angebunden.
Das Navigieren auf der Karte wurde mit einem Home-Button und einem FullScreen Plugin ergänzt.
Die einzelnen Todesfälle sowie die geclusterten CircleMarker wurden durch Popups mit Details ergänzt, die durch Anklicken aktiviert werden.
Zur Darstellung der Fälle pro County kamen verschiedene Ansätze in Frage. Zuerst wurde überlegt, die Informationen zu den einzelnen Counties ausgehend von Polygonen zu visualisieren, wozu ein entsprechender Datensatz von Connecticut Open Data im Projekt eingebunden wurden. Doch letztlich hat sich die Umsetzung mit CircleMarker als praktischer erwiesen. Dafür wurden entsprechende zentrale Koordinaten aus Wikipedia/GeoHack gesammelt und in einer Variable in einem separaten Javascript File angelegt.
Die Erarbeitung der anderen HTML-Seiten, vor allem das Sammeln von Inhalt, lief parallel zur Gestaltung der interaktiven Karte. Besonders nach dem Befüllen der Informations-Seiten wurde das Styling

des Contents eine Aufgabe, die sich bis zum Ende des Projekts durchgezogen hat. Von der Schriftart, über Icons im Tab sowie in der Karte und im Fließtext, bis zur Positionierung der Grafiken, wurde kontinuierlich am Aussehen der Seite gefeilt, um ein konsistentes und ansprechendes Interface zu kreieren. Der Feinschliff der Webseite gelang mittels Modal-Funktion, um die Grafiken durch Anklicken vergrößert darzustellen. Dazu wurden zusätzliche Dateien (modal.css und modal.js) erstellt, um die Übersichtlichkeit zu bewahren. Auch der Header wurde schlussendlich separat gestyled und mittels einem eigenen header.css angesprochen.

Abbildung 4 gibt abschließend noch einen Überblick über die Informationen, die in den einzelnen Datensätzen gesammelt sind, zusammen mit dem Index des Arrays.

| Index | Content | Index | Content | Index | Content |
|---|---|---|---|---|---|
| **8** | ID | **22** | InjuryPlace | **36** | Benzodiazepine |
| **9** | Date | **23** | InjuryCity | **37** | Ethadone |
| **10** | DateType | **24** | InjuryCounty | **38** | Amphet |
| **11** | Age | **25** | InjuryState | **39** | Tramad |
| **12** | Sex | **26** | COD | **40** | Morphine_NotHeroin |
| **13** | Race | **27** | OtherSignifican | **41** | Hydromorphone |
| **14** | ResidenceCity | **28** | Heroin | **42** | Other |
| **15** | ResidenceCounty | **29** | Cocain | **43** | OpiateNOS |
| **16** | ResidenceState | **30** | Fentanyl | **44** | AnyOpioid |
| **17** | DeathCity | **31** | FentanylAnalogue | **45** | MannerofDeath |
| **18** | DeathCounty | **32** | Oxycodone | **46** | DeathCityGeo |
| **19** | Location | **33** | Oxymorphone | **47** | ResidenceCityGeo |
| **20** | LocationifOther | **34** | Ethanol | **48** | InjuryCityGeo |
| **21** | DescriptionofInjury | **35** | Hydrocodone | **49** | TownIndex |

*Abbildung 4: Übersicht über die Daten mit Indexangabe für Array*

# 3. Besondere Elemente der Seite

Nachdem nun der Aufbau des Projekts allgemein vorgestellt wurde, wird nun auf besondere Elemente der Seit noch genauer eingegangen.

## 3.1. Interaktive Karte

### 3.1.1. Allgemeine Beschreibung und Übersicht zu Plugins

Die Karte zur Darstellung der Todesfälle nach Überdosis ist interaktiv gestaltet und bietet grundsätzlich zwei verschiedene Visualisierungen der Daten an: genauere Informationen zu den einzelnen Todesfällen und die Anzahl von Todesfällen pro County. Zum Einbinden der Basemaps wurde auf verschiedene Leaflet Provider zurückgegriffen. Für eine bessere Interaktion kann die Karte auch im Vollbildmodus angezeigt (Plugin: Fullscreen) und zurück zum definierten Zentrum der Karte navigiert werden (Plugin: ZoomHome).

**Informationen zu den einzelnen Todesfällen (Abbildung 5):** Details zu den einzelnen registrierten Fällen werden in Popups visualisiert, welche an Marker gebunden sind, positioniert am jeweiligen Todesort. Es gibt Anzeigeoptionen für alle Fälle oder nach Geschlecht, wobei jeweils nur eine Option ausgewählt und visualisiert werden kann (Plugin: Grouped Layer Control). Die Option „No Selection" wurde eingebaut, um die individuellen Todesfälle auch ausblenden zu können (Anm.: eine entsprechende Option im Plugin wurde nicht gefunden). Um die große Anzahl von Datenpunkten (5105) performancetechnisch überhaupt darstellen zu können, mussten diese geclustert werden (Plugin: Marker Cluster). Dies ermöglicht dem Nutzer außerdem eine bessere Übersicht.

**Funktionen:**

- *drawAccidents (datapoints, layer):* Zeichnen von Markern mit Popup für jeden Todesfall
- *filterdata (data, index, key):* Filtern von Dateneinträgen
- *drawAccidentsFemale():* Visualisierung von weiblichen Todesfällen
- *drawAccidentsMale():* Visualisierung von männlichen Todesfällen

**Anzahl der gesamten Fälle pro County (Abbildung 6):** mit der Visualisierung von Kreisen mit relativem Radius und angehängtem Popup kann die Anzahl von Fällen in allen acht Counties verglichen werden. Die Kreise sind am geographischen Zentrum des jeweiligen Counties positioniert. Ein besonderes Feature auf der Webseite ist die zeitliche Entwicklung von 2012 bis 2018, für welche in einer Animation die Anzahl der Fälle pro Monat dargestellt wird.

**Funktionen:**

- *CountyCount (data):* Zählen der Fälle pro County
- *drawCountyCount (ArrayWithCountyCounts, layer, sizefactor, popuptext):* Zeichnen von Kreisen für jedes County proportional zur Anzahl der Todesfälle
- *CountyCountsPerMonth(data_raw):* Zählen der Fälle pro County pro Monat
- *playButton.onclick()*: Zeitlicher Verlauf der Anzahl der monatlichen Fälle von 2012 bis 2018

**Übersicht über verwendete Plugins**

- MarkerCluster*:* Durch den relativ großen Datensatz wurde das Laden der einzelnen Fälle, besonders der Koordinaten der einzelnen Fälle durch den MarkerCluster verbessert.
- Grouped Layer Plugin: Auswahl eines einzelnen Layers in der Control der Karte
- ZoomHome Plugin: Zoomt zurück zum Ausgangspunkt/Zentrum der Karte
- FullScreen Plugin: Ermöglicht Ansicht der Karte in Vollbildmodus

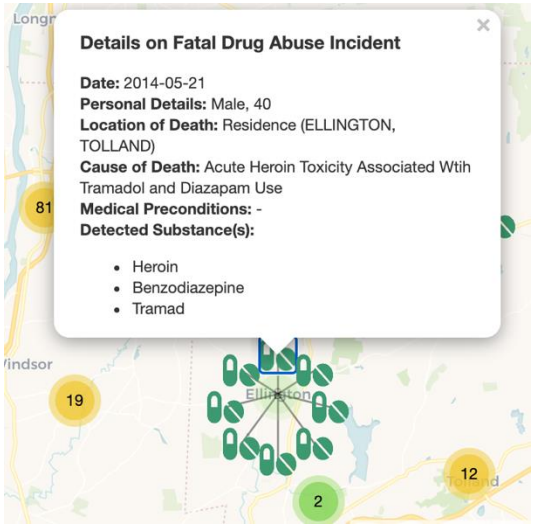


*Abbildung 5: Informationen zu den einzelnen Todesfällen*

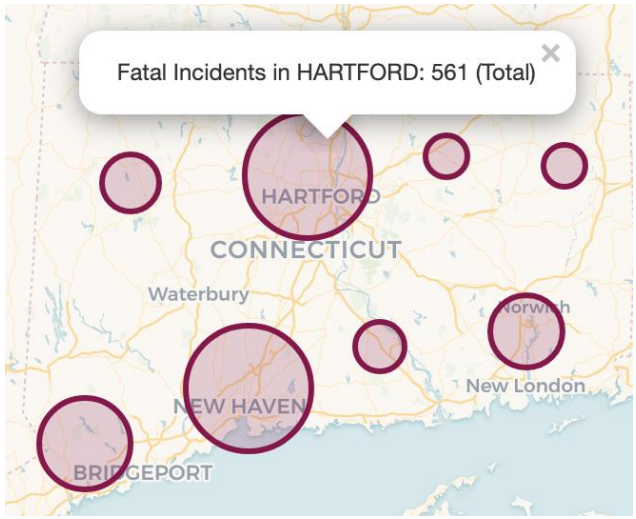


*Abbildung 6: Anzahl der gesamten Fälle pro County*

### 3.1.2. Beschreibung der Funktionen

**Zeichnen von Markern mit Popup für jeden Todesfall:** *drawAccidents (datapoints, layer):*

Die Funktion platziert individualisierte Marker am Sterbeort (Koordinaten in Index [46[1], 46[2]]). Dazu werden Informationen der einzelnen Dateneinträge abgefragt (Datum [9], Geschlecht [12], Alter [11], Ort des Todes [17-20], Todesursache [26], Medizinische Vorerkrankungen [27], Nachgewiesene Substanzen [28-44]) und im Popup angezeigt. Für die Abfrage der nachgewiesenen Drogen muss dabei auf die Metadaten zugegriffen werden, da im Dateneintrag bei einem positiven Test auf die bestimmte Droge nur ein „Y“ vermerkt ist. Um die nachgewiesenen Drogen abzufragen, werden die entsprechenden Indizes 28-44 in einem Array gesammelt, über welchen anschließend iteriert werden kann. Um für die Abfrage der Metadaten den richtigen Index anzusprechen, muss auf den Index „i“ 28 addiert werden. Für den Fall „Other“ Drugs ([42]) wird eine gesonderte Abfrage ausgeführt, da die Information hierbei in den Daten selbst und nicht den Metadaten steht. Sämtliche Informationen im Popup reagieren darauf, ob die entsprechenden Variablen im jeweiligen Eintrag definiert sind oder nicht. Der Ziellayer für den Overlay ist als Parameter angegeben, wodurch die Funktion sehr flexibel nutzbar ist.

```
//Abfrage der nachgewiesenen Drogen
//Teste auf bestimmte Substanzen in element[28] bis element[44] - Sammeln in Array
        let DetectDrugs = [];
        for (let index = 28; index < 45; index++) {
            DetectDrugs.push(element[index]);
        }
        let substances = []; //Sammeln der nachgewiesenen Substanzen
        for (let i = 0; i < DetectDrugs.length; i++) { //Index 0 startet mit element[28]
            let testDrug = DetectDrugs[i];
            let index = 28 + i; //Key für Element - gleich für DATA.data und DATA.meta
            let substance;

            if (testDrug == "Y") { //wenn Droge nachgewiesen ("Yes")
                substance = DATA.meta.view.columns[index].name; //Name der Droge aus Metada-
ten abrufen
                substances.push(substance)
            }
        };
```

```
        if (element[42] !== null) { //spezielle Abfrage für "Other"
            substance = `Other: ${element[42]}`
            substances.push(substance)
        }
        let detectSubst = substances.join("</li><li>"); //Zusammenfügen der Dro-
gen in String bzw. Template für unsortierte Liste
```

**Filtern von Dateneinträgen:** *filterdata (data, index, key):*

Diese Funktion loopt durch den Datensatz und filtert die Einträge durch das Prüfen von einem bestimmten Wert (key) in einem bestimmten Index. Die gefilterten Daten werden in einen Array übergeben und beim Aufrufen der Funktion in einer Variable gespeichert. Für dieses Projekt wurde der Filter nur für weibliche und männliche Fälle angewandt, doch die Möglichkeiten, die sich dadurch zur Visualisierung der Daten ergeben, gehen natürlich darüber hinaus.

→ Anwenden in `drawAccidentsFemale()` & `drawAccidentsMale()`

```
//Filter Funktion
let filterData = function (data, index, key) {
    let datalist = [];
    for (let i in data) {
        if (!data.hasOwnProperty(i)) continue;
        let element = data[i];

        if (element[index] == key) {
            datalist.push(element)
        }
    }
    return datalist;
}
```

```
//Female Cases
let drawAccidentsFemale = function () {
    let dataFemale = filterData(DATA.data, 12, "Female");
    drawAccidents(dataFemale, overlay.drugaccidents_female);
}
drawAccidentsFemale();
```

**Zählen der Fälle pro County:** *CountyCount (data):*

Um die Fälle für jedes County zu zählen, werden zunächst jeweils eine Zählvariable mit dem Wert 0 initialisiert. Anschließend kategorisiert die *switch* Funktion die Daten nach dem Sterbeort (County) [18]. Für jeden positiven Fall wird die entsprechende Zählvariable um den Wert 1 erhöht. Nach abgeschlossener Kategorisierung werden die Zählvariablen in einen Array übergeben, der mit Ausführen der Funktion in einer Variable gespeichert wird.

**Zeichnen von Kreisen für jedes County proportional zur Anzahl der Todesfälle:** *drawCountyCount (ArrayWithCountyCounts, layer, sizefactor, col, popuptext):*

Diese Funktion zeichnet basierend auf dem Array mit den gezählten Fällen für jedes County einen Kreis, dessen Radius über den Wert der Zählvariable definiert wird. An den Kreis ist auch ein Popup gebunden, in welchem auch angegeben ist, welcher Zeitraum in diesem Moment visualisiert wird. Die Koordinate

des Kreiszentrums wird aus der Variable `county_center` abgerufen, welche aus GeoHack ermittelt wurden. Eine Bedingung für diese Funktion ist die gleiche Reihenfolge der Counties im Array der Zählvariablen und Koordinaten.
Die Größe des Kreises wird von einem Größenfaktor kontrolliert, der, ebenso wie der Ziellayer und Farbe des Kreises als Parameter definiert ist. Hintergrund ist hierbei, dass die Gesamtanzahl der Fälle und die Fälle pro Monat weit auseinander liegen, sodass keine einheitliche Skalierung sinnvoll ist. Außerdem werden bei Zählvariablen mit dem Wert 0 keine Kreise gezeichnet. Eine wichtige Komponente ist außerdem der Befehl `clearLayers()`, welcher bei einem mehrmaligen Aufrufen der Funktion das Überzeichnen von Kreisen verhindert.

```
const county_center = [
    ["HARTFORD", 41.81, -72.73],
    ["NEW HAVEN", 41.35, -72.9],
    ["FAIRFIELD", 41.23, -73.37],
    ["NEW LONDON", 41.47, -72.1],
    ["LITCHFIELD", 41.79, -73.24],
    ["MIDDLESEX", 41.44, -72.52],
    ["WINDHAM", 41.83, -71.99],
    ["TOLLAND", 41.85, -72.33],
]
```

**Zählen der Fälle pro County pro Monat:** *CountyCountsPerMonth(data_raw):*

Damit die Anzahl der Todesfälle pro County pro Monat visualisiert werden kann, müssen diese zunächst gezählt werden. Im ersten Schritt sortiert die Funktion dazu die Fälle des Datensatzes nach dem Datum [9]. Dateneinträge ohne Datum (*null*) müssen ignoriert werden, um die Sortierung nicht zu stören. Da das Datum im Datensatz in einem ISO Format dokumentiert wurde, kann mit der Initialisierung `new Date` auf bestimmte Elemente des Datums zugegriffen werden, für diesen Fall auf Jahr (`getFullYear()`), Monat (`getMonth()`) und der kombinierten Information Jahr-Monat. Außerdem werden Arrays zum Sammeln von Dateneinträgen pro Monat angelegt.

```
    //Sortieren des Datensatzes nach Todesdatum
    data.sort(function (row1, row2) {
        let date1, date2;
        date1 = row1[9];
        date2 = row2[9];
        if (date1 < date2) {
            return -1;
        } else if (date1 > date2) {
            return 1;
        }
        return 0;
    });

    //Einzelne Monate abfragen
    let collectMonth = []; //Sammelt alle Dateneinträge eines Monats
    let collectAllCountsPerMonth = []; //Sammelt alle CountyCounts pro Monat
```

```
   //Zeilenweiser Vergleich von Einträgen, um Monate zu differenzieren
   for (let i = 1; i < data.length; i++) {
       let element1 = data[i - 1]; //Eintrag 1 wird mit Eintrag 2 verglichen
       let element2 = data[i];

     //Datum im Datensatz ist in ISO Format gespeichert --> Abruf mit "new Date"
        let date1 = new Date(element1[9]);
        let date2 = new Date(element2[9]);

        let year1 = date1.getFullYear(); //Jahr
        let month1 = date1.getMonth() + 1; //Monat (Hintergrund von +1: Month In-
dex ist 0-baisert (Januar = 0))
        let YearMonth = date1.toISOString().substring(0, 7); //Information Monat

        let year2 = date2.getFullYear(); //Jahr
        let month2 = date2.getMonth() + 1; //Monat (Hintergrund von +1: Month In-
dex ist 0-baisert (Januar = 0))

        //ID für jeden Monat aus Summe von "Jahr" und "Monat" gebildet
        //Dadurch können Monate voneinander unterschieden werden
        let monthID1 = year1 + month1; //z.B. Januar 2012 = 2013
        let monthID2 = year2 + month2; //z.B. Februar 2012 = 2014

        //Vergleich der Dateneinträge
        if (monthID1 == monthID2) { //solange gleiche "MonatID" (ergo glei-
cher Monat)
            collectMonth.push(element1);
            continue; //Nächstes "Datenpaar" abfragen

        } else { // = Monatsgrenze (element1 in anderem Monat als element2)
            collectMonth.push(element1); //letzter Eintrag des Monats

            let countMonth = CountyCount(collectMonth); //Zählen der Todes-
fälle pro County pro Monat
            collectAllCountsPerMonth.push([YearMonth, countMonth]);
           collectMonth = []; //Array leeren für nächsten Monat
       };
   };
```

Das Differenzieren von einzelnen Monaten basiert auf dem zeilenweisen Vergleich von zwei Einträgen (element1 und element2) über eine MonatsID, die aus der Summe von Jahr und Monat gebildet wird. Einzelne Dateneinträge werden so lange dem Array „`collectMonth`“ zugefügt, bis die MonatsID des Datenpaares nicht mehr gleich ist, was zum Monatswechsel der Fall ist (z.B. Januar 2012 (`monthID1 = 2013`) und Februar 2012 (`monthID2 = 2014`)). Dann wird der dann letzte Eintrag des Monats dem Array noch angehängt, bevor die Funktion *CountyCount(collectMonth)* zum Zählen der Todesfälle für diesen Monat ausgeführt wird. Der dadurch erstellte Array „countMonth“ wird dem finalen Array „collectAllCountsPerMonth“ zusammen mit der Information Jahr-Monat hinzugefügt. Anschließend muss der Array „`collectMonth`“ in Vorbereitung für die nächste monatliche Zählung wieder geleert werden.

## 3.2. Slider und Animation

Um die Anzahl der Fälle in jedem County für jeden Monat zu visualisieren, wurde sich bei der Implementierung des Sliders und Animation (Play / Stop) am CODVID-Beispiel des Webmapping Kurses orientiert. Der Slider wird hierbei über den Array „`AllCountsPerMonth`“ (Ergebnis von der Funktion „*`CountyCountsPerMonth`“)* definiert, dessen einzelne Indizes und Werte über die Position des Sliders abgerufen werden. Sobald sich die Position des Sliders verändert, wird die Funktion `drawCountyCount` mit der entsprechenden monatlichen Zählung `([slider.value][1])` ausgeführt.

Zum Starten der Animation ist neben dem Slider ein Play-Stop-Button positioniert. Außerdem wird das jeweilige Jahr-und Monat zur Position des Sliders und damit visualisiertem Monat rechts des Buttons dargestellt. Diese Information wird auch in das Popup übergeben. Die Funktion zur Animation mit Start- und Pausiermöglichkeit erhöht den Wert des Sliders iterativ um eins, womit zum nächsten Monat gesprungen werden kann. Die Kreise werden auch gezeichnet, wenn die Animation zu einem bestimmten Punkt pausiert wird.

index.html

```
<div id="map"></div>
<div id="sliderDiv">
     <input type="range" id="slider">
     <input id="play" type="button" value="▶">
     <div id="dateOutput"></div>
</div>
```

Main.js

```
let dateOutput = document.querySelector("#dateOutput")

slider.onchange = function () {
    overlay.accidents_county.clearLayers();
    overlay.accidents_county_month.addTo(map);
    let index = slider.value;
    let dataMonth = AllCountsPerMonth[index][1]
    let month = AllCountsPerMonth[index][0];
    dateOutput.innerHTML = month;

    drawCountyCount(dataMonth, overlay.accidents_county_month, 200, "#39CCCC",
month);
};
```

## 3.3. Navigation mit Hover

Die Gestaltung der Navigationsleiste wurde so vorgenommen, dass sowohl bei voller Fensterbreite, wie auch bei einem schmaleren Browserfenster, diese voll funktionsfähig ist, ohne, dass einzelne Elemente nicht mehr angezeigt werden, bzw. in die nächste Zeile verschoben werden.

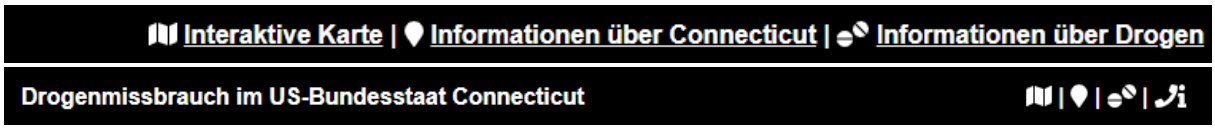


*Abbildung 7: Verschiedene Ansicht der Navigation abhängig von der Browserbreite*

Hierfür wurde im Header der HTML-Files eine schmale und eine breite Ansicht eingerichtet und jeweils einer Klasse zugewiesen. Für alle vier Verlinkungen wurde wie folgt vorgegangen:

```
<div class="winmin">
    <a href="index.html" style="text-decoration: none; ">
        <i class="minmap"><i class="fas fa-map"></i></i>
        <div class="hide1">Karte</div>
    </a> |
```

Darauf aufbauend wurde folgender CSS-Code für den Wechsel der beiden Ansichten und der Hover-Funktion geschrieben:

```
@media screen and (max-width: 1529px){
    div.winmax {
        display:none;
    }
}
@media screen and (min-width: 1530px){
    div.winmin {
        display:none;
    }
}
.minmap:hover + .hide1 {
    display: inline-block;
}
.hide1 {
    display: none;
}
```

Der Wechsel zwischen den beiden Ansichten findet ab einem bestimmten Pixel-Schwellenwert (1530 px) statt. Mit Hilfe des Hovers wird dem Nutzer beim Überfahren der Symbole mit dem Curser angezeigt, auf welche Seite er gelangen kann. Auch hier wurde für alle vier Elemente der Hover, sowie die Hide-Funktion eingerichtet.

## 3.4. Vergrößerung der Bilder mit Modal

Für alle Bilder der Website soll es die Möglichkeit geben, diese beim Anklicken größer anzeigen zu lassen. Hierfür wurde die grundlegende Codestruktur von w3schools: How TO - Modal Images verwendet. Für eine übersichtlichere Darstellung und Ordnung der Daten wurde der CSS- und JS-Code in neue Files ausgelagert, welche mit den HTML-Seiten verknüpft wurden. Hierbei wurde festgestellt, dass ein Laden des JS-Skriptes im head nicht funktioniert, weshalb das Skript am Ende des Bodys eingefügt wurde.

Darüber hinaus musste neben dem CSS- und JS-Code noch folgender HTML-Code zur Einbettung des Modals ergänzt werden:

```
<div id="myModal" class="modal">
    <span class="close">&times;</span>
    <img class="modal-content" id="img01">
    <div id="caption"></div>
</div>
```

In einem ersten Implizierungsschritt sollte nur ein Bild die Modal-Funktion erhalten. Als jedoch auf einer einzelnen HTML-Seite mehrere Bilder die Modal-Funktion erhalten sollten, reichte eine ID, wie in der Ursprungsversion des Codes, nicht mehr aus und der JS-Code musste für die gewünschte Anwendung wie folgt umgeschrieben werden:

```
let modal = document.getElementById("myModal");
let modalImg = document.getElementById("img01");
let captionText = document.getElementById("caption");
 ('zoomImg');
for (let i = 0; i < array.length; i++) {
    array[i].onclick = function () {
        modal.style.display = "block";
        modalImg.src = this.src;
        captionText.innerHTML = this.alt;
    }
}
let span = document.getElementsByClassName("close")[0];
span.onclick = function () {
    modal.style.display = "none";
}
```

Grundlegende Bausteine, wie das Modal und Modal-Image blieben dabei erhalten. Ebenso die Funktion für das Schließen des Modals. Da auf Grund der Umwandlung der ID in eine Class sich nun mehrere Elemente darin befinden, wurde eine Schleife eingebaut, die die onclick-Funktion des Modals für jedes einzelne Element ausführt. Dadurch ist es nun möglich beliebig viele Bilder mit Hilfe der Zuweisung `class="zoomImg"` vergrößert darzustellen.

## 3.5. Styling der HTML Seiten

Das Styling der HTML-Seiten belief sich auf viel Feintuning und Abstimmen von Content innerhalb der einzelnen Seiten, abgesehen von den in den vorigen Punkten bereits beschriebenen Features.
Die verwendeten Icons für das Tab, in der Karte und bei den Bildunterschriften stammen von FontAwesome. Die verwendete Font *Quicksand* wurde mittels

```
<link href=https://fonts.googleapis.com/css2?family=Quicksand&display=swap
rel="stylesheet">
```

eingebaut.

Abgesehen vom main.css wurde für den Header (und das Modal) jeweils ein zusätzliches .css erstellt. Besonders für die im Header eingearbeitete Funktion, die den Text zu den Icons in der Navigation aus- und einblendet, war ein zusätzliches Stylesheet zur Übersichtlichkeit nützlich.
Um die verwendeten Grafiken mit passenden Bildunterschriften zu versehen wurde statt `img` ein `<figure>` Element eingebaut, um den Zusammenhang zu den Bildern herzustellen.

```
<figure>
    <img class="zoomImg" src="images/nationaldrugoverdose2018-graph2.jpg" alt="Fatale Unfälle aufgrund von Drogenmissbrauch in den USA von 1999 bis 2018" style="width:100%;max-width:400px">
    <figcaption></figcaption><i class="fas fa-camera"></i> Fatale Unfälle aufgrund von Drogenmissbrauch in den USA von 1999 bis 2018 <br />
    <a href="https://www.drugabuse.gov/drug-topics/trends-statistics/overdose-death-rates">&copy; National Institute on Drug Abuse</a>
    </figcaption>
</figure>
```

Die unter dem Punkt “Todesfälle durch Drogenkonsum in Connecticut” nebeneinander dargestellten Grafiken werden im info_region.html per div und class angesprochen

```
<div class="box1"> <img src="images/Diagramm_Alter.PNG" style="width:100%"> Anzahl der Todesfälle nach Alter </div>
```

und im main.css dann entsprechend gestyled.

```
.box1, .box2, .box3 {
    float: left;
    width: 32%;
    margin-right: 2%;
    padding: 5px;
    background: white;
    box-sizing: border-box;
}
.box3 {
    margin-right: 0;
}
```

Durch die Verwendung der Grafikboxen können die einzelnen Elemente leichter angesprochen und nebeneinander platziert werden.

## 4. Herausforderungen und offene Probleme

Es wurde bereits angesprochen, dass sich mit der Größe des Datensatzes zu Beginn des Projekts Schwierigkeiten in der Visualisierung ergeben haben. Dies lag daran, dass die Datenpunkte jeweils einzeln

dargestellt worden sind, was häufig zum Aufhängen der Seite geführt hat. Dieses Problem konnte jedoch sehr gut mit der geclusterten Darstellung gelöst werden. Allerdings bleibt hier noch der Nebeneffekt, dass sich an manchen Orten, besonders Krankenhäusern, die Datenpunkte häufen, was nach wie vor etwas unübersichtlich ist (Abbildung 8).

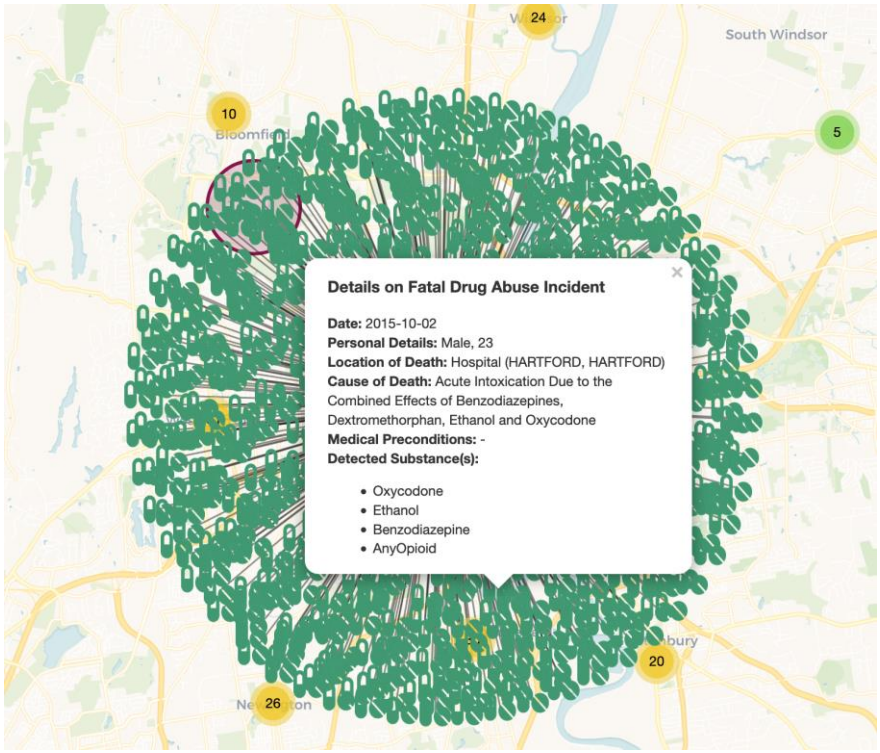


*Abbildung 8: Häufung von Datenpunkten*

Ein mittelgroßes Fragezeichen steht zudem noch bei der Datenlücke im Array der monatlichen Zählungen pro County im Jahr 2016, in welchem keine Fälle registriert wurden, obwohl es diese definitiv in diesem Zeitraum gibt. Der Grund dafür ist allerdings noch unklar.

Ein aktuelles Problem ist derzeit noch, dass nach Aktivieren der Sliderfunktion bzw. Animation und dem Aufrufen des Layers mit den monatlichen Kreiszeichnungen („by month“), der Layer mit den fällen („Total (2012-2018)“), nicht mehr aktiviert werden kann. Dies kann möglicherweise bis zur Präsentation noch behoben werden.

Außerdem gibt es scheinbar in manchen Browsern noch Probleme bei dem Ausrichten des Kartenausschnitts an den visualisierten Layer, weswegen die Funktion `mapfitbounds ()` möglicherweise nicht richtig implementiert wurde. Schlussendlich gibt es natürlich immer noch Spielraum, die Darstellung der Informationen in Popups, z.B. die Großschreibung der Counties zu verfeinern. Außerdem ist im Hinterkopf zu behalten, dass das Abrufen des Datums über `new Date` nicht die lokale Ortszeit bzw. das lokale Ortsdatum ausgibt, weswegen etwa die monatlichen Grenzen leicht verschoben sind, was sich jedoch nicht großartig negativ auswirkt.

Bezüglich der Anpassung der Elemente in verschiedenen Browsern mit unterschiedlicher Auflösung oder Fenstergröße fällt auf, dass sich teilweise die Hover-Elemente im Header oder die Jahr-Monats-Angabe am Slider verschieben und nicht mehr sinnvoll genutzt bzw. ansprechend dargestellt werden.

# 5. Ausblick

Die Beschäftigung mit dem Datensatz zu den Todesfällen aufgrund von Überdosis im Staat Connecticut war im Projekt durchweg sehr spannend, da sehr vielschichtige Informationen bereitgestellt werden. Gleichzeitig ist uns natürlich bewusst, dass es sich hierbei um ein sehr ernstes Thema und tragische Einzelschicksale handelt, die visualisiert werden, weswegen es uns mit der Webseite auch wichtig ist, für das Thema zu sensibilisieren und Hintergrundinformationen bereitzustellen.

Für die Visualisierung der Karte bestehen darüber hinaus mit der angelegten Filterfunktion und den flexibel einsetzbaren Funktionen natürlich noch eine Vielzahl weiterer möglicherer Darstellungen der Daten. Beispielsweise könnte der Anwender der Karte auch eine Altersrange für die dargestellten Einzelfälle manuell festlegen oder nach anderen persönlichen Daten filtern. Auch die zeitliche Animation gibt Spielraum für die Visualisierung der Entwicklung weiterer thematischen Layer. Schließlich gibt es für die Hälfte des Datensatzes auch Koordinaten nicht nur für den Ort des Todes, sondern auch Wohnungsort und Ort der Verletzung, welche auch eine räumliche Nachverfolgung des Unfallhergangs erlaubt.

# Quellen

**Datenquellen:**

State of Connecticut (2019): Accidental Drug Related Deaths 2012-2018. Local Government, Health and Human Services. Verfügbar unter: https://catalog.data.gov/dataset/accidental-drug-related-deaths-january-2012-sept-2015 [abgerufen am 16.06.2020]

State of Connecticut (2019a): Accidental Drug Related Deaths 2012-2018. Local Government, Health and Human Services. Verfügbar unter: https://data.ct.gov/d/rybz-nyjw/visualization [abgerufen am 16.06.2020]

**Plugins:**

MarkerCluster: https://cdnjs.com/libraries/leaflet.markercluster
Grouped Layer Plugin: https://github.com/ismyrnow/leaflet-groupedlayercontrol
ZoomHome Plugin: https://github.com/torfsen/leaflet.zoomhome
FullScreen Plugin: https://github.com/Leaflet/Leaflet.fullscreen

**Quellen Content:**

Wikia (2020). Drogen Wikia. Verfügbar unter: https://drogen.wikia.org/de/wiki/Drogen_Wiki:Hauptseite [abgerufen am 16.06.2020]

Wikipedia (2020). Connecticut. Verfügbar unter: https://de.wikipedia.org/wiki/Connecticut [abgerufen am 14.06.2020]

Wikipedia (2020a). Drogen. Verfügbar unter: https://de.wikipedia.org/wiki/Droge [abgerufen am 16.06.2020]

Connecticut's Official State Website (2020). General Description and Facts. Verfügbar unter: https://portal.ct.gov/About/General-Description-and-Facts [abgerufen am 14.06.2020]

United States Census Bureau (2019). American Community Survey: Demographic and Housing Estimates. Verfügbar unter: https://data.census.gov/cedsci/table?q=Connecticut&g=0400000US09&hidePreview=true&tid=ACSDP1Y2018.DP05&table=DP05 [abgerufen am 14.06.2020]

United States Census Bureau (2019a). QuickFacts Conecticut. Verfügbar unter: https://www.census.gov/quickfacts/fact/table/CT/BZA110217#BZA110217 [abgerufen am 14.06.2020]

Statista (2020). Unemployment rate in Connecticut from 1992 to 2019. Verfügbar unter: https://www.statista.com/statistics/412963/unemployment-rate-in-connecticut-since-1992/ [abgerufen am 14.06.2020]

U.S. Department of Commerce (2020). Gross Domestic Product (GDP) summary, quarterly by state. Bureau of Economic Analysis. Verfügbar unter: .https://apps.bea.gov/iTable/iTable.cfm?reqid=70&step=1&isuri=1&acrdn=1#reqid=70&step=1&isuri=1&acrdn=1[abgerufen am 14.06.2020]

National Institute on Drug Abuse (2020). National Drug Overdose Deaths - Number Among All Ages, by Gender, 1999-2018. Verfügbar unter: https://www.drugabuse.gov/drug-topics/trends-statistics/overdose-death-rates [abgerufen am 14.06.2020]