Robert Orthofer, BSc

Browserseitig erstellte Visualisierung für geotemporale multivariate Daten am Fallbeispiel Mapoch

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

Masterstudium Geospatial Technologies

eingereicht an der

Technischen Universität Graz

Betreuer

Dr. Konrad Rautz

Institut für Geodäsie

Graz, Juni 2018

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

KURZFASSUNG

Die vorliegende Masterarbeit liefert einen Überblick über den Status Quo der Visualisierungsmethoden geotemporaler Daten mit besonderen Fokus auf Anwendungen im Web. Hierzu werden durch einen interdisziplinaren Ansatz Teilbereiche der Kartographie, Diagrammdarstellung, Datendesign, Geoinformatik und Webentwicklung näher beleuchtet und miteinander vereint. Dies geschieht unter anderem durch die Analyse standardisierter Webtechnologien und deren Eignung zur Darstellung klassischer sowie moderner Grafiken aus den Bereichen der Diagrammdarstellung und der Kartographie. Als praktischer Teil der Arbeit werden die Erkenntnisse der Arbeit in der Webapplikation *Mapoch* umgesetzt und als Machbarkeitsstudie präsentiert. *Mapoch* soll über eine clientseitige Dateneinspeisung sowohl für Laien als auch für Experten eine Möglichkeit darstellen, große geotemporale Datensätze ohne Upload auf einen Server interaktiv zu visualisieren.

ABSTRACT

This master thesis shows an overview of the status quo of methods to visualize geotemporal data, with a special focus on web applications. This is done by following an interdisciplinary approach, highlighting and fusing insights from cartography, chart design, data design, geoinformatics and web development. Standardized web technologies are analyzed for their fitness for use concerning the creation and display of classic and modern diagrams and cartographic illustrations. In the practice part of this thesis the web application *Mapoch* is created as an implementation of the insights of the theoretical part. *Mapoch* is a pilot study which provides a suitable option to interactively visualize large geotemporal datasets for both non-professionals and professionals alike, while keeping the data local without a server upload.

Um die Lesbarkeit zu erleichtern, wird in der Arbeit die Sprachform des generischen Maskulinums verwendet. Ich weise darauf hin, dass die Verwendung dieser Form geschlechtsneutral zu verstehen ist.

INHALT

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

VERZEICHNIS DER CODEBEISPIELE

ABKÜRZUNGSVERZEICHNIS

API	Application Programming nterface
ASI	Austrian Standards Institute
CRS	Coordinate Reference System
CSS	Cascading Style Sheets
CSV	Comma-separated Values
D3	Data-Driven Documents
DOM	Document Object Model
DPI	Dots per Inch
ECMA	European Computer Manufacturers Association
EPSG	European Petroleum Survey Group
ESRI	Environmental Systems Research Institute
GeoJSON	Geographic JavaScript Object Notation
GIF	Graphics Interchange Format
GIS	Geographic Information System
GML	Geography Markup Language
GPKG	GeoPackage
GPU	Graphics Processing Unit
HSL	Hue Saturaton Lumination
HTML	Hypertext Markup Language
Http	Hypertext Transfer Protocol
ISO/TC	Internationale Organisation für Normung / Technical Comittee
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
MIME	Multipurpose Internet Mail Extensions
MIT	Massachusetts Institute of Technology
npm	Node Package Manager
OGC	Open Geospatial Consortium
ON	Österreich Norm
OSM	Open Street Map
OWS	OGC Web Service
PDF	Portable Document Format
PNG	Portable Network Graphics
RGB	Red Green Blue
SHP	Shapefile
SRID	Spatial Reference System Identifier
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
VML	Vector Markup Language
W3C	World Wide Web Consortium
WCS	Web Coverage Service
WFS	Web Feature Service
WGS	World Geodetic System
WMS	Web Map Service
WMTS	Web Map Tile Service
XML	Extensible Markup Language

1 EINLEITUNG

1.1 Die neue Welt der Karten

Karten waren früher etwas Besonderes. Man benutzte Karten zur Orientierung im Gelände, in einer unbekannten Stadt oder bereits als Vorbereitung auf die lange Autofahrt dorthin. Man las Karten in Atlanten, sowohl topographische Karten ganzer Kontinente sowie thematische Karten über die Bevölkerungsentwicklung und Wirtschaft einzelner Bundesländer. Karten waren äußerst komplexe graphische Darstellungen, welche Unmengen an Informationen nicht nur räumlich verorteten, sondern oft auch inhaltlich miteinander verbanden. Man setzte sich hin und nahm sich Zeit für die vielschichtige Tätigkeit des Kartenlesens.

Heute sind Karten und ähnliche Darstellungen ubiquitär. Jeder benutzt sie, oft nur beiläufig, notwendige Informationen müssen sofort sichtbar sein. Jede kleinste Unstimmigkeit kann zu falschen Interpretationen der Karten führen oder, noch schlimmer, zu dessen Ablehnung. Kartographen müssen sich heute anderer Probleme bewusst sein wie vor 20 Jahren, denn die Art und Weise, wie Karten vom Betrachter wahrgenommen werden, hat sich grundlegend geändert.

Die Hauptursache dafür sind die rapiden technologischen Fortschritte sowie die steigende Verfügbarkeit dieser Technologien der letzten Jahre und damit einhergehend die aufstrebende Akzeptanz der Menschen gegenüber Computer und dem Internet.

Im Internet existiert eine Vielzahl verschiedener Arten von Karten. Einige sind interaktiv und man kann Informationen der Karte abfragen, welche bei einer statischen Karte nicht erhältlich sind. Andere sind animiert und zeigen Veränderungen mit der Zeit, andere sind haptisch und man ‘fühlt‘ Texturen und Karten mit Ton können dem Benutzer Informationen erzählen. (nach TYNER 2015, S. 4)

Die technologischen Entwicklungen im Computerbereich stellen Kartographen, Geoinformatiker und Softwareentwickler also nicht nur vor neue Probleme, wie man korrekt Informationen über Karten im Internet übermittelt, sondern liefern uns gleichzeitig eine noch viel überwältigendere Anzahl an Möglichkeiten, wie man diese Probleme adressieren kann.

1.2 Ziel der Arbeit

Im Rahmen dieser Masterarbeit möchte der Autor auf einige dieser neuen Möglichkeiten eingehen. Als Forschungsobjekte dienen zwei Problemstellungen der klassischen Kartographie: die Darstellung von geotemporalen Daten sowie die Darstellung von multivariaten Geodaten. Um die Fähigkeiten der digitalen

clientseitigen Kartographie im Browser zu demonstrieren, wird versucht, diese beiden Probleme zu kombinieren und in Einklang zu bringen.

Zu weiteren Aufgaben gehören die Erreichbarkeit der verschiedensten Endnutzer, der Unklarheit über deren Vorwissen sowie die allgemeine Sicherheit der Daten. Um dies zu erreichen, soll als Endprodukt dieser Masterarbeit eine Webapplikation entstehen, welche eine clientseitige Visualisierung der oben genannten kartographischen Aufgabenstellungen ermöglicht.

1.3 Methodik

Grundlage dieser Arbeit sind ausführliche Literatur- und Internetrecherche sowie die Analyse bereits vorhandener Applikationen und Software in diesem Gebiet. Die Literatur besteht zum Teil aus kartographischen Standardwerken und Hochschulschriften, zum anderen aus Fachliteratur der Softwareentwicklung, Informationsdesign, Mediendesign und Kommunikation mittels Grafik und Gestaltung.

Durch die Analyse von vorhandenen verwandten Applikationen sollen sowohl bestehende Probleme als auch positive Umsetzungen identifiziert werden. Anschließend soll auf dem erarbeitenden Wissen eine eigene Applikation zur clientseitigen Webisualisierung von geotemporalen multivariaten Daten umgesetzt werden. Diese basiert auf einer bereits vorhandenen vom Autor erstellten Webapplikation, welche von Andreas Hocevar und Johannes Leitner auf dem Symposium für angewandte Geoinformation AGIT 2016 unter dem Titel „Es lebt – Mit Drag&Drop von Zeitreihen räumlicher Daten zur Kartenanimation im Browser“ als Produkt vorgestellt wurde.

In dieser Masterarbeit wird eine Auswahl sinnvoller Softwareoptionen behandelt, allerdings soll das Endprodukt ausschließlich auf Open Source Software basieren sowie selbst auch als Open Source Produkt zugänglich sein. Unterschiede zwischen existierender Open Source und kommerzieller Software werden als Exkurse kurz erläutert.

2 DATENVISUALISIERUNGSANSÄTZE IM WEB

Meistens liegen Daten im Web beziehungsweise die Ergebnisse von Datenanalysen in numerischer Form vor. Der Mensch hat jedoch ein verhältnismäßig schlechtes Vermögen, die überwältigenden Informationen aus raumzeitlichen Datensätzen in Text- und Zahlenform im Kopf zu visualisieren und zu erfassen. Eine Visualisierung von multidimensionalen Geodaten durch einen intuitiven 3D-Effekt kann diesen Prozess erleichtern, potentielle Muster und Zusammenhänge komplexer Sachverhalte können leichter erkannt werden. (vgl. YANG et al 2015, S. 342ff) Der Zugang zu den Daten ist natürlicher, anstelle eines Vergleichs von Tabellen können multidimensionale Daten „erlebt“ werden.

Nach BURKHARD (2006, S. 202) besitzen visuelle Repräsentationen von Informationen folgende Vorteile:

- Auslösen von Emotionen (zum Beispiel Werbung)
- Aufzeigen von Beziehungen (zum Beispiel Diagramme)
- Sichtbar machen von Ausnahmen, Trends oder Muster (zum Beispiel Sternbild)
- Erregen und halten der Aufmerksamkeit (Kino)
- Erinnerungshilfen (zum Beispiel Metaphern)
- Gleichzeitig einen Überblick und Details darstellen (zum Beispiel Landkarte)
- Erleichterung des Lernens (zum Beispiel Funktionsschemas)
- Hilfe bei Koordination in Teams (zum Beispiel Fußballstrategieszenarien)
- Motivation und Aktivierung von Menschen (zum Beispiel Visionen)

Es sollte jedoch Abstand davon genommen werden, alle quantitativen Daten automatisch als visuelle Repräsentation darzustellen. Einer Darstellung der Daten sollte eine Abwägung der Bedürfnisse des Benutzers vorhergehen. Nach FEW (2012, S. 51) haben tabellarische Darstellungen und Diagramme unterschiedliche Anwendungsgebiete. Seiner Meinung nach sollten Werte möglicherweise dann in einer tabellarischen Form dargestellt werden, wenn eine oder mehrere der folgenden Kriterien erfüllt wird (FEW 2012, ebd.):

1. die Darstellung wird zum Nachschlagen einzelner Werte benutzt
2. sie wird dazu genutzt, einzelne Werte miteinander zu vergleichen, jedoch nicht zum Vergleich von Serien von Werten
3. es werden präzise Werte benötigt
4. die zu übermittelnden quantitativen Informationen liegen in mehr als einer Maßeinheit vor
5. es werden sowohl Zusammenfassungen der Daten sowie Detailwerte verlangt

Nach FEW (2012, S. 51ff) liegt die Stärke von grafischen Darstellungsformen in der Übermittlung von Informationen, die in der Form der Werte versteckt liegen, wie etwa

in Muster, Trends oder Ausnahmen in den Daten. Die Darstellung von Informationen in einer Grafik macht demnach nicht automatisch die bedeutsamen Muster in den Daten besser sichtbar als in einer Tabelle. Die Grafik muss so gestaltet werden, dass sie das Phänomen untermauert, welches mit den Daten erzählt werden soll.

2.1 Diagrammtypen

Hinsichtlich der oben genannten Vorteile von visuellen Repräsentationen versucht man, mit Diagrammen Beziehungen aufzuzeigen, Trends und Muster erkennbar zu machen sowie einen Überblick über die Datengrundlage als Gesamtheit zu erschaffen. Grundsätzlich können alle klassischen Diagrammtypen aus Printmedien auch für eine Datenvisualisierung im Web verwendet werden. Die Vielzahl an Möglichkeiten, sich Diagramme mit On- und Offlineprogrammen schnell erstellen zu lassen, kann zu einer unüberlegten Wahl der Diagrammart führen. Im diesem Kapitel werden einige gängige Diagrammtypen mit möglichen Anwendungsbereichen vorgestellt.

2.1.1 BAR CHART

Bar Charts, Column Charts oder Säulendiagramme eignen sich zur Darstellung von komparativen Daten klassifiziert in ein nominales oder ordinales Skalenniveau. Bar Charts können vertikal oder horizontal dargestellt werden, möchte man negative Daten darstellen wird eine vertikale Darstellung empfohlen, da die meisten Menschen intuitiv negativ mit „unten“ assoziieren.[1]

Da wir in der westlichen Kultur den Zeitverlauf von links nach rechts denken, kann bei vertikalen Säulendiagrammen (also vertikale Säulen) der Eindruck einer Zeitreihe entstehen. Ist dies in den Daten nicht der Fall, ist meist ein horizontales Diagramm zu bevorzugen. (vgl. ZELANY 2001, S. 35)

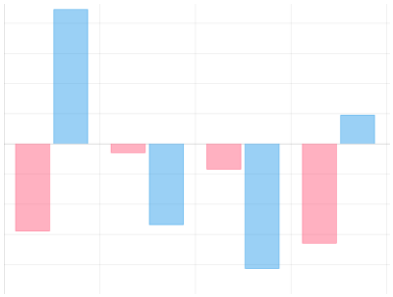


Abbildung 1 Vertikale Bar Chart (Quelle: chartjs.org)

[1] CHARTBLOCK (2017): When to use a bar Chart. https://www.chartblocks.com/en/support/faqs/faq/when-to-use-a-bar-chart

2.1.2 LINE CHART

Zeitreihen können entweder durch Column Charts oder durch Line Charts dargestellt werden.
Line Charts eignen sich bei einer größeren Anzahl an Untergliederungen, oder wenn eine Veränderung gezeigt werden soll, also wenn ein Wert von einem zum nächsten Zeitpunkt weitergetragen wird, wie etwa bei Bestandsdaten. Im Unterschied dazu eignen sich Column Charts bei geringerer Anzahl an Untergliederungen und wenn bei jedem Zeitschritt der Wert wieder bei null beginnt, wie etwa bei Produktionsdaten oder bei Verkehr. (nach ZELANY 2001, S. 36)

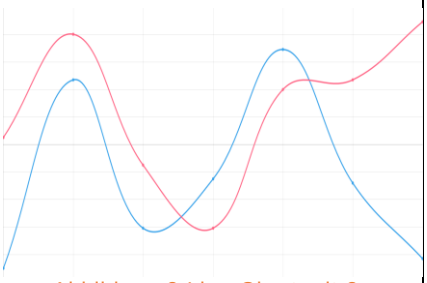


Abbildung 2 Line Chart mit 2 Datensätzen (Quelle: chartjs.org)

2.1.3 PIE CHART (NACH ZELANY 2001, S. 28FF)

Ein Vergleich zwischen Komponenten eines einzelnen Elements kann durch ein Pie Chart (Kreisdiagramm) erzeugt werden. Ein Kreis erzeugt den Eindruck einer Ganzheit, daher eignet sich ein Pie Chart ausschließlich zur Darstellung von Prozentensätzen eines Ganzen.
Für Pie Charts sollten maximal 6 Kategorien benutzt werden, wobei die sechste Kategorie eine Zusammenfassung aller weiteren Kategorien unter „Sonstige“ darstellen kann.
Ein explodieren, also auseinanderbrechen der einzelnen Elemente wird nicht empfohlen, da dieser Effekt keinen Mehrwert zur Information erzeugt und die einzelnen Segmente nur noch schlechter vergleichbar werden[2].

Abbildung 3 Pie Chart (Quelle: chartjs.org)

[2] EEA (2016): Chart dos and don’ts. https://www.eea.europa.eu/data-and-maps/daviz/learn-more/chart-dos-and-donts

2.1.4 DONUT CHART

Donut Charts sind im Prinzip nichts Anderes als Kreisdiagramme mit einem Loch in der Mitte. Die Aufgabengebiete sind exakt dieselben wie jene der herkömmlichen Kreisdiagramme, jedoch wird die Identifikation der einzelnen Komponenten sowie die Wahrnehmung der jeweiligen Prozentsätze unnötig erschwert. Bei Donut Charts können anstelle von Winkeln und Flächen nur die Länge der gebogenen Linien miteinander verglichen werden. (NUSSBAUMER KNAFLIC 2015 , S. 65)

In der Literatur wird von dieser Darstellungsform meist komplett abgeraten, ein einziger Pluspunkt ist der Freiraum in der Mitte, der in seltenen Fällen als Platz für Labels genutzt werden kann, wenn ein Labeling außerhalb nicht möglich ist.

Abbildung 4 Donut Chart (Quelle: chartjs.org)

2.1.5 BUBBLE CHART

Unter “Bubble-Charts” versteht man eine Spezialform der Scatter-Charts zur Darstellung multivariater Daten. Neben zwei Variablen der X- und Y-Achse kann eine weitere Variable durch die Größe der Punkte dargestellt werden. Wichtig dabei ist die Wiedergabe eines Wertes durch den Flächeninhalt der Kreise, da das menschliche Auge natürlicherweise die Größen und nicht die Radien oder Durchmesser vergleicht. Eine vierte Variable kann durch den Einsatz von Farbe eingebunden werden.[3] In Abbildung 5 wird so zum Beispiel der jeweilige Quadrant farblich hervorgehoben.

In der Geographie findet diese Art der Darstellung multivariater Daten häufige Anwendung, wobei zwei Variablen meist die geographische Lage eines Objektes wiedergeben. Dies hat jedoch häufig auch den Vorteil, dass Überschneidungen der Kreise selten sind beziehungsweise durch grafische Hilfsmittel vermieden werden können.

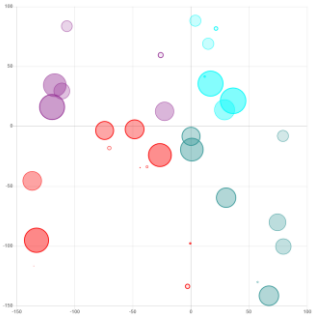


Abbildung 5 Bubble Chart (Quelle: chartjs.org)

Der Gebrauch von 3D-Darstellungen in Diagrammen ist immer zu vermeiden, es sei denn, es wird eine echte dritte Dimension in den Daten dargestellt und selbst dann kann es sehr schnell zu inakzeptabel komplizierten Grafiken führen. Dreidimensionalität als Verzierung von Diagrammen kann die Daten bis zur Interpretationsunmöglichkeit verzerren. Zusätzlich führt 3D zu andernfalls unnötigen

[3] STARR, B. (2015): How to Design Bubble Charts. https://visage.co/data-visualization-101-bubble-charts/

Elementen wie Seiten- und Bodenpanele, Farbverläufe, Schatten, etc. (NUSSBAUMER KNAFLIC 2015, S. 65ff).

Dies steht in einem leichten Widerspruch zur Kommunikation im Webdesign. Möchte der Designer mitteilen, dass ein Element klickbar ist, gibt es dafür einige Möglichkeiten. Die Diagramme an sich sollten möglichst schlicht gehalten werden, bei einigen Diagrammen kann jedoch durch einen leichten Schlagschatten auf den Untergrund eine Illusion der Schwebe erzeugt werden. Schwebende Objekte erzeugen das Gefühl, dass man sie hinunterdrücken, also mit ihnen interagieren kann[4]. Dieser Effekt ist besonders bei mobilen Anwendungen interessant, bei denen zusätzliches Feedback über Mouse-Hovering nicht möglich ist.

Abschließend soll erwähnt werden, dass die Aufgabe von Diagrammen die Vereinfachung von Daten in Textform, wie etwa Tabellen, für den Betrachter ist. Sind diese Grafiken mühsamer zu entschlüsseln als die Ausgangsdaten, so haben sie ihren Zweck verfehlt. Besonders leicht geschieht dies bei zusammengesetzten Diagrammen aus mehreren Grundformen, häufig eine Unumgänglichkeit bei einer multivariaten Datengrundlage.

Ein großer Vorteil von simpleren Diagrammen in der Kartographie ist die einfachere grafische Verortung im Vergleich zu Text und Tabellen. Mehr dazu im Kapitel 4.4: Darstellung geotemporaler Daten: Status Quo).

2.2 Webbasierte Visualisierung von Daten ohne Raumbezug

Der Großteil der Datenvisualisierung im Web geschieht über die beiden offenen W3C-Standards Canvas und SVG, beziehungsweise über Bibliotheken, die sich dieser Technologien bedienen. Diese Standards sind in allen gängigen Browsern implementiert und benötigen somit, im Unterschied zu Adobe Flash oder Microsoft Silverlight, keinerlei weiteren Plugins.

In diesem Kapitel wird näher in die offenen Standards Canvas und SVG eingegangen, es werden Bibliotheken vorgestellt, Anwendungsgebiete aufgezeigt sowie die Vor- und Nachteile der jeweiligen Technologien diskutiert.

2.2.1 CANVAS-BASIERTE VISUALISIERUNG

Canvas ist ein HTML5-Element zur Generierung und Manipulation von Bitmap-Graphiken im Browser. Der Zugriff erfolgt mittels JavaScript über die 2D Context API. Canvas funktioniert nach dem “immediate mode”-Prinzip, während Flash, Silverlight und SVG nach dem “retained mode”-Prinzip funktionieren. Das bedeutet, Canvas rendert nach jeder Veränderung die gesamte Szene neu, während die anderen

[4] KLIEVER, J. (2015): Designing With Light and Shadow: 10 Highly Effective Tips You Should Try [With Case Studies] https://designschool.canva.com/blog/light-and-shadow/

Technologien die Szene als abstrakte Vektorformen zwischenspeichern und später darauf wieder zugreifen können (siehe Abbildung 6). Als Entwickler hat man durch das “immediate mode”-Prinzip jedoch mehr Kontrolle über low-level-Verfahren.[5]

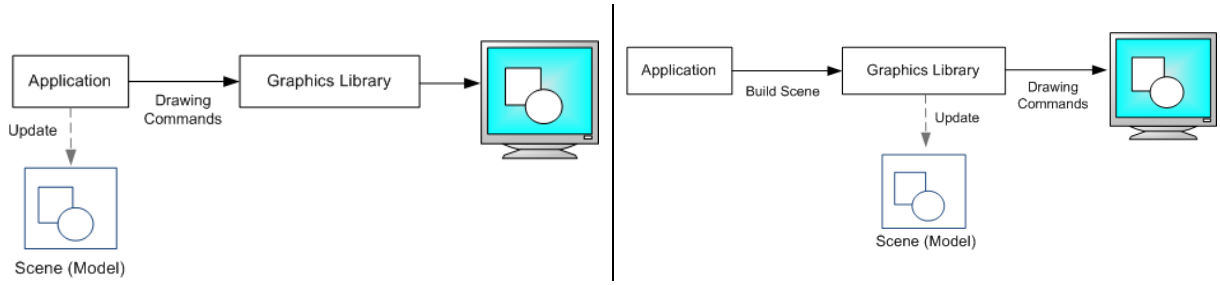


Abbildung 6 immediate-mode graphics (links) und retained mode graphics (rechts) (Quellen: msdn.microsoft.com)

Code 1 zeigt den HTML-Code zur Erstellung eines Rechtecks und eines darüber liegenden Kreises. Die Formen müssen mit JavaScript über die 2D Context API erstellt werden. Hierzu muss zuerst über den Befehl `HTMLCanvasElement.getContext()` ein Zeichenkontext für das jeweilige HTML5 Canvas Element erhalten werden.

Code 1 Beispiel Canvas Quadrat und Kreis. (Quelle: eigene Abbildung)

```
<!DOCTYPE html>
<html>
  <body>
    <canvas id="myCanvas" width="100" height="100"></canvas>
    <script>
      var c = document.getElementById("myCanvas");
      var ctx = c.getContext("2d");
      ctx.fillStyle = "black";
      ctx.fillRect(0, 0, 100, 100);
      ctx.fillStyle = 'red';
      ctx.arc(50,50,40,0,2*Math.PI);
      ctx.fill();
    </script>
  </body>
</html>
```

Es ist auch möglich, Grafiken wie Diagramme über JavaScript oder entsprechender Bibliotheken zu erstellen und dann in einer Webkarte als Marker für Point-Features zu benutzen. OpenLayers erlaubt für das `ol.style.Icon` –Objekt für Vector Features ein solches `HTMLCanvasElement`[6]. Für Leaflet existieren Plugins der Community um dies zu ermöglichen.[7]

[5] FULTON, J. (2013): HTML5 Canvas. O'Reilly Media, 2. Edition http://proquest.techbus.safaribooksonline.de/book/web-development/html/9781449335847/1dot-introduction-to-html5-canvas/ch01_html?uicode=tugraz
[6] OpenLayers (2017): ol.style.Icon https://openlayers.org/en/latest/apidoc/ol.style.Icon.html
[7] KAVUN, S. (2016): L.CanvasIcon https://github.com/sashakavun/leaflet-canvasicon

2.2.1.1 Chart.js

Chart.js ist eine open-source Javascript-Bibliothek zur Erstellung von Diagrammen in einem HTML5 Canvas. Chart.js ist unter der MIT-License frei verfügbar.[8]

Chart.js ermöglicht die Erstellung folgender vorgefertigter Diagrammtypen:

- Bar Chart
- Line Chart
- Area Chart
- Scatter
- Donut
- Pie
- Polar Area
- Radar
- Bubble

Bei allen Diagrammen sind Farben, Labels, Animationen, Tooltips und on-event JavaScript Funktionen möglich. Die meisten Diagramme haben weitere Individualisierungsmöglichkeiten, wie etwa das Tauschen der X- und Y-Achse, Stapelung (Bar- und Area Charts), Interpolationen, Gruppierungen und logarithmische Skalen.

Chart.js verfolgt die Philosophie, möglichst einfach funktionale und performante interaktive Diagramme für das Web zu erstellen. Der Funktionsumfang ist im Vergleich zu anderen Bibliotheken eher gering, die oben genannten Standard-Diagramme können jedoch mit geringer Einarbeitungszeit erstellt werden. Chart.js benötigt auch kein externes Rendering, die Diagramme werden direkt im Browser über HTML5-Standards erstellt.

2.2.1.2 Vega & Vega-Lite

Vega ist eine Visualisierungsgrammatik zur Erstellung von interaktiven Diagrammen und Grafiken. Vega-Lite ist eine noch höherliegende Grammatik basierend auf Vega, sie ist kompakter und zugänglicher. Vega-Lite wird auf Vega zurückkompiliert und ist abhängig von D3 (siehe Kapitel 2.2.2.2). Alles in Vega und Vega-Lite geschieht im JSON-Format. Die Erstellung der Grafiken geschieht in R und sie werden in einem `canvas` gerendert. Dieses kann in eine Website übergeben werden.[9]

Vega beziehungsweise Vega-Lite verfügen über eine Vielzahl von vorgefertigten Diagrammtypen, inklusive Heatmaps, Network Diagrams und Choroplethenkarten.

[8] Chart.js (2017): http://www.chartjs.org/docs/latest/
[9] RUDIS, B. (2016): Create Vega-Lite specs & widgets with the vegalite package https://rud.is/b/2016/02/27/create-vega-lite-specs-widgets-with-the-vegalite-package/

Abbildung 7 zeigt eine Spezialform der Bubble Chart, erstellt mit Vega-Lite. Diese Diagrammart visualisiert zum Beispiel die Anzahl der Commits in einem GitHub-Repository.

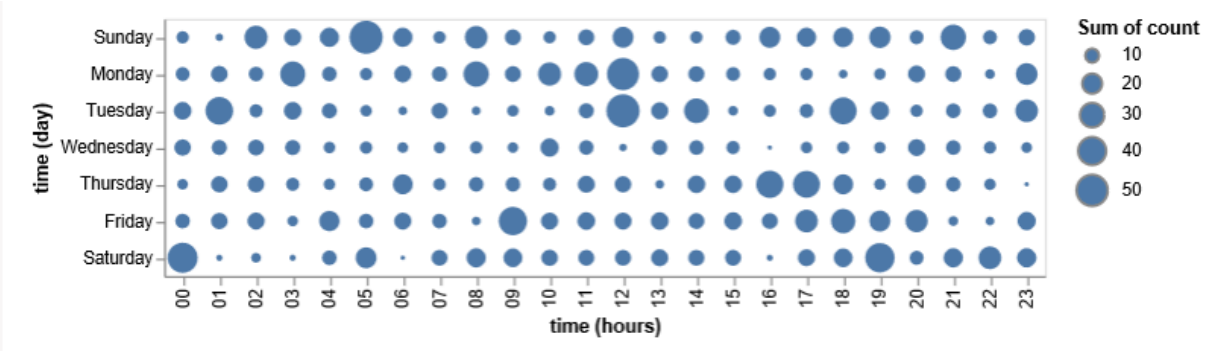


Abbildung 7 Table Bubble Plot in Vega-Lite (Quelle: vega.github.io)

2.2.2 SVG-BASIERTE VISUALISIERUNG

SVG (Scalable Vector Graphic) ist eine XML-basierte Möglichkeit, 2D-Graphiken im Web zu definieren. Durch die XML-Struktur wird jede Form als Objekt im SVG-DOM gespeichert. Ändert sich ein Attribut, so wird die Form neu gerendert.[10] Code 2 zeigt ein simples Beispiel der SVG-XML-Struktur. Es wird kein JavaScript benötigt.

Code 2 Beispiel SVG Quadrat und Kreis (Quelle: eigener Code/eigene Abbildung)

```
<!DOCTYPE html>
<html>
  <body>
    <svg width="100" height="100">
      <rect width="100" height="100"/>
       <circle cx="50" cy="50" r="40"fill="red" />.
    </svg>
  </body>
</html>
```

Wie bereits in Kapitel 2.2.1: Canvas-basierte Visualisierung beschrieben, kann auf das SVG-DOM jederzeit von außerhalb zugegriffen werden. So können über CSS und JavaScript Werte verändert werden, Animationen definiert und Interaktionen wie Event Handlers hinzugefügt werden.

In den folgenden Kapiteln werden einige JavaScript-Bibliotheken vorgestellt, welche den Umgang mit SVG erleichtern sollen, indem sie vorgefertigte Diagramme bereitstellen oder die Art und Weise, wie auf SVG-Objekte zugegriffen werden kann, vereinfachen.

2.2.2.1 Highcharts

Highcharts ist eine JavaScript Charting Library basierend auf SVG in neueren Browsern, VML in älteren Browsern und Canvas in Android 2.x. Auch Highcharts liefert

[10] W3CSchools (2017): HTML5 SVG. https://www.w3schools.com/html/html5_svg.asp

eine große Anzahl an vorgefertigten Diagrammtypen, zusätzlich werden Möglichkeiten zur Kombination der Diagrammtypen und zur Vorprozessierung der Daten bereitgestellt. [11] Highcharts ist nach Chart.js die zweitmeist über npm heruntergeladene JavaScript Charting Library[12], diese Statistik inkludiert jedoch nicht D3, da D3 nicht als reine Charting Library gehandelt wird.

Highcharts ist nur frei verfügbar für nicht-kommerzielle Nutzung, also für Non-Profit Organisationen oder für den privaten Gebrauch.[13]

2.2.2.2 D3

D3 (Data-Driven Documents) ist eine JavaScript-Bibliothek zur datenabhängigen DOM-Manipulation über die W3C-Webstandards SVG, HTML und CSS. Das Ziel von D3 ist nicht, vorgefertigte Diagramme bereitzustellen, welche nur mehr mit Daten gefüttert werden müssen, sondern es widmet sich der Wurzel des Problems, der effizienten Manipulation der Dokumente, abhängig von Daten.[14]

Durch die Variabilität von D3 ist die Erstellung praktisch aller vektorbasierten Visualisierungen inklusive Animationen und Interaktionen möglich. D3 weist eine aktive Community auf, welche auch für eine Vielzahl von verschiedenen online zugänglichen Beispielgrafiken verantwortlich ist. Neben den klassischen Diagrammen existieren auch viele sehr spezielle Visualisierungen, welche ein entweder sehr limitiertes Anwendungsgebiet aufweisen oder in der Praxis überhaupt nicht zu empfehlen sind. Viele diese Grafiken dienen auch rein als „Eye Catcher“.

Abbildung 8 zeigt eine Spezialform des Donut Charts. Eine mögliche Anwendung für diese Visualisierungsform ist die Darstellung der Häufigkeit, wie oft ein Navigationsweg in einer Website gewählt wurde[15]. Über einen Mouse-Hover Effekt werden alle anderen Segmente, außer jene des dazugehörigen Navigationpfades, farblich abgeschwächt. Der Freiraum innerhalb der Kreissegmente wird, wie in Kapitel 2.1.4: *Donut Chart* erwähnt, zur Darstellung des jeweiligen Zahlenwertes beziehungsweise für weitere Informationen genützt.

Das eigentliche Label der Segmente wird dynamisch oberhalb der Grafik erstellt. Der Benutzer hat keine Möglichkeit, ohne Interaktion mit der Grafik, die Segmente zu

---

[11] SHAHID, B. (2014): Highcharts Essentials. http://proquest.techbus.safaribooksonline.de/book/databases-and-reporting-tools/9781783983964/1dot-getting-started-with-highcharts/ch01lvl1sec08_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODE3ODM5ODM5NjQlMkZjaDAxbHZsMXNlYzA4X2h0bWwmcXVlcnk9

[12] BAAJ, A. (2017): Compare the Best Javascript Chart Libraries of 2017. https://blog.sicara.com/compare-best-javascript-chart-libraries-2017-89fbe8cb112d

[13] Highcharts (2017): Licenses. https://shop.highsoft.com/highcharts

[14] BOSTOCK, M. (2017): Data-Driven Documents. https://d3js.org/

[15] RODDEN, K. (2017): Sequences sunburst (d3 v4). https://bl.ocks.org/kerryrodden/766f8f6d31f645c39f488a0befa1e3c8

identifizieren. Die Grafik muss also vom Benutzer aktiv erforscht werden, um diese Informationen zu erhalten. Dies ist bei wissenschaftlichen Diagrammen eindeutig ein Nachteil, bei anderen Anwendungsgebieten kann dieses erzwungene Erforschen des Benutzers durchaus gewollt sein.

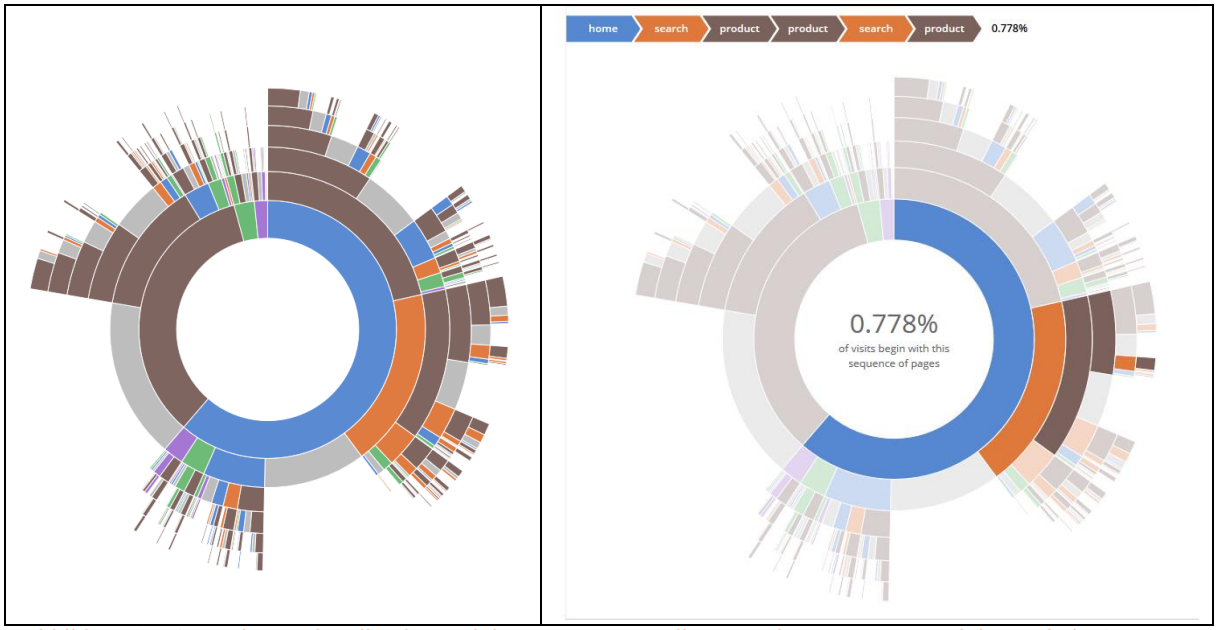


Abbildung 8 D3 "sunburst visualization", Links: Ausgangsstadium, Rechts: Hover-Interaktion auf ein Segment (Quelle: bl.ocks.org)

Da keine Diagramme an sich vorgegeben werden, sondern D3 nur eine andere Möglichkeit ist, Webdokumente zu manipulieren, ist die generelle Meinung über D3, dass es ein mächtiges Tool für effiziente Grafiken ist. Allerdings wird dies von einer steilen Lernkurve sowie einem generell größeren Zeitaufwand begleitet, da jedes Diagramm eine Einzelanfertigung ist.

D3 eignet sich auch zur Erstellung von vektorbasierten Webkarten. Mehr dazu im Kapitel 3.3.1: *D3*.

2.3 Canvas oder SVG?

Tabelle 1 zeigt die technischen Überlegungen des W3C zu SVG und der 2D Context API, wobei das HTML5 `<canvas>`-Element als Zugriff zu Zweiterem fungiert.[16] Canvas ist script-based, es ist dafür gemacht, Objekte dynamisch über JavaScript zu erstellen, während SVG für einen Dokument-basierten Einsatz entworfen wurde. Canvas verfolgt einen Raster- beziehungsweise Pixel-basierten Ansatz, SVG ist ein Format zur Darstellungen von Vektoren. Die 2D Context API ist eine low-level API, während SVG einen Scene-Graph besitzt, also ein Modell mit allen geometrischen Primitiven und

[16] W3.org (2011): HTML/Canvas and SVG. https://www.w3.org/community/webed/wiki/HTML/Canvas_and_SVG

Events. Auf dieses Modell kann auch nach dem Rendering der Graphik sowohl mit CSS als auch mit JavaScript zugegriffen werden.

Die Performance der Elemente ist abhängig von der Anwendung. HTML5 Canvas bietet eine bessere Performance bei einer größeren Anzahl von Objekten, insbesondere wenn diese häufig neu gezeichnet werden müssen. SVG ist performanter bei weniger Objekten und bei einer größeren Rendering-Fläche.[17]

Tabelle 1 Technische Überlegungen des W3C zu Canvas und SVG (Quelle: w3.org)

2D Context API	SVG
script-based	document-based
pixel operations	vectors
low-level graphics API, no scene graph	object model with events
2D Context API	accessible
fast rendering	performance hit with large numbers of shapes

In Tabelle 2 werden die Intentionen des W3C von Canvas und SVG genauer erläutert. Einhergehend mit den oben erläuterten technischen Merkmalen eignet sich Canvas für Applikationen mit vielen sich ändernden Objekten wie Spiele, jedoch auch für dynamische Webkarten und andere Visualisierungen. SVG ist gedacht für User Interfaces und einfachere Grafiken mit interaktiven Events und Animationen. Der Raster-Ansatz von Canvas eignet sich besser für komplexe Bitmap-Bilder, der Vektor-Ansatz von SVG bietet eine uneingeschränkte Skalierbarkeit der modellierten Objekte.

Durch das Dokumenten-basierte XML-Format von SVG kann dies auch in externen Programmen wie Adobe Illustrator, Inkscape oder CorelDRAW erstellt und online über CSS verändert werden. Dadurch ist dieses Format besonders für Webdesigner gedacht. Canvas-Grafiken müssen über JavaScript erstellt werden, durch die größere Kontrolle der low-level-API sind sie somit besser für Entwickler geeignet.

Tabelle 2 Beabsichtigte Verwendungszwecke des W3C von Canvas und SVG (Quelle: w3.org)

2D Context API	SVG
fast-paced games	user interfaces
highly dynamic visualizations	interactive animations
very complex images	scalable images
for developers	for designers

Die Wahl der geeignetsten Technologie für die Datenvisualisierung ist also von der Anwendung abhängig, allerdings muss erwähnt werden, dass in der Praxis die Wahl

[17] SMUS, B. (2009): Performance of canvas versus SVG. http://smus.com/canvas-vs-svg-performance/

oft auch durch Pragmatismus entschieden wird. Häufig ist in einem Projekt bereits eine der Technologien in Verwendung, der Einsatz weiterer Bibliotheken könnte die Ladezeiten negativ beeinflussen. Auch die Einarbeitungszeit in eine andere Technologie beziehungsweise Bibliothek kann entscheidend sein, insbesondere dann, wenn die Veränderung der Performance nur eine untergeordnete Rolle spielt.

3 WEBBASIERTE VISUALISIERUNG VON GEODATEN

3.1 Typen geographischer Web-Tools

Nach Mitchel existieren verschiedene Arten digitaler Mapping-Tools. Er unterteilt sie in fünf verschiedene Kategorien, wobei diese sich häufig überlappen (nach MITCHEL 2005, S. 17ff):

- Viewing and Mapping
- Analysis
- Creating and Manipulating
- Conversion
- Sharing

3.1.1 VIEWING AND MAPPING

Unter „Viewing“ versteht man das Betrachten von Karten, unter Mapping die Kartenerstellung. Beide Arten beinhalten meist einen graphischen Output. Beispiele für Viewingtools sind allgegenwärtige Online-Kartendienste wie Google Maps, Bing Maps oder Open Street Map. Ein Tool zum Erstellen von Karten ist zum Beispiel der Carto-Editor von CartoDB, in dem eigene Geodaten dargestellt oder auf eine Bibliothek von bereits vorhandenen Daten zugegriffen werden kann. Die erstellten Karten sind individualisierbar, können gespeichert, als Bild exportiert (siehe Abbildung 9) oder als interaktive Webkarte eingebunden werden[18].

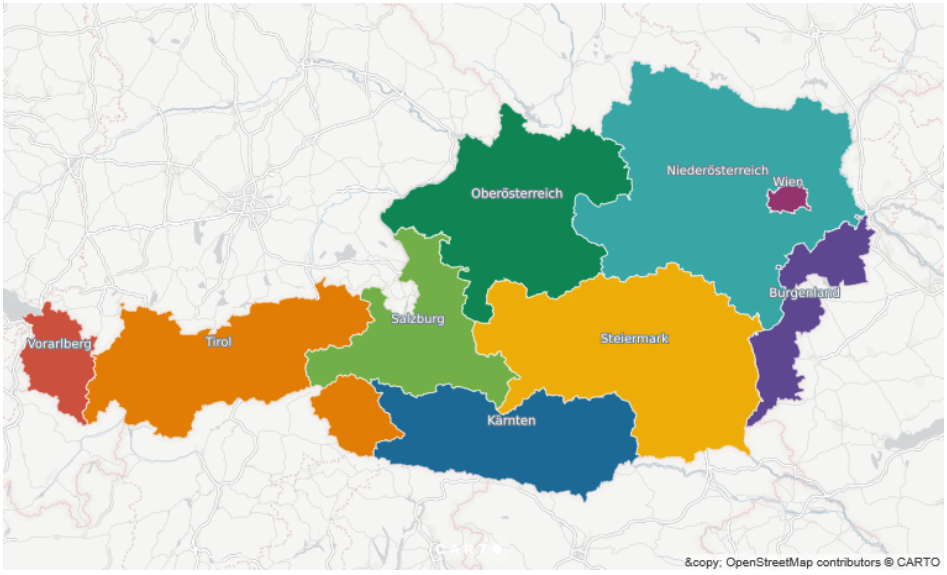


Abbildung 9 Die Bundesländer Österreichs, erstellt mit CartoDB mit eigenen Geodaten (Landesgrenzen und Name) und einer Basemap von Carto, basierend aus OSM-Daten (Quelle: carto.com)

[18] CARTO (2017): CARTO Editor Workflow. https://carto.com/docs/carto-editor/workflow/

3.1.2 ANALYSIS

Analysetools können eigenständige Web-Tools darstellen, häufig werden Analysen auch als Preprocessing für ein verständliches Kartenprodukt benötigt. Hierzu zählen unter anderem Klassifikationen, geographische Filterung sowie statistische Auswertungen der Ausgangsdaten. Auch webbasierten Routingdiensten geht eine Analyse eines Straßengraphs voraus, das Ergebnis ist eine geeignete Route, eventuell mit zusätzlichen Informationen wie voraussichtlicher Fahrtzeit, Höhenprofil und Abbiegefolge (siehe Abbildung 10).

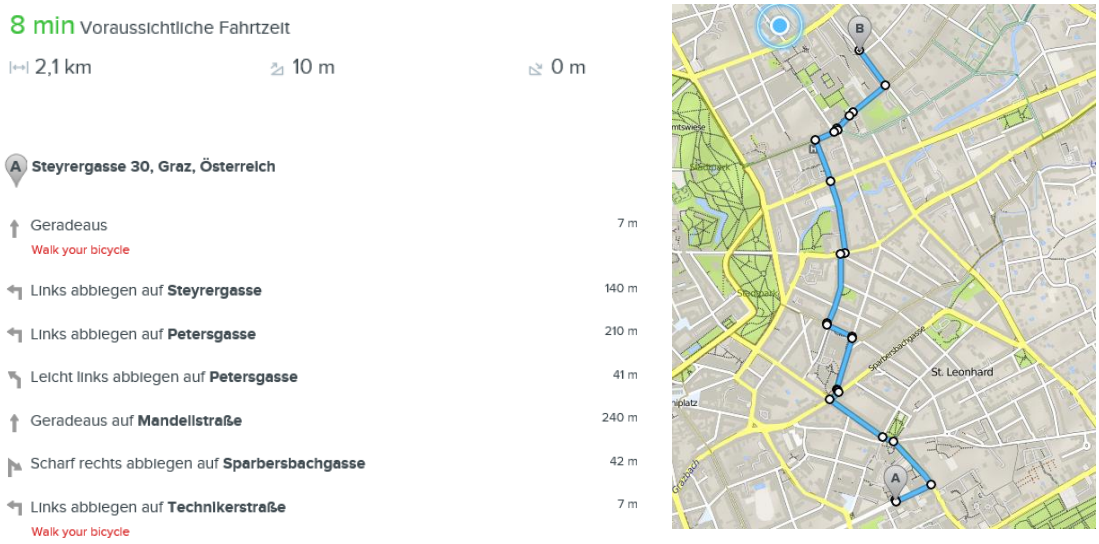


Abbildung 10 Online Routing und Guidance, das Ergebnis der Analyse eines Straßengraphens. Quelle: bikemap.net

3.1.3 CREATING AND MANIPULATING

Creating ist die Erstellung neuer Features, unter Manipulation versteht man das permanente Verändern von Daten. Es können sowohl die Lage und die Form von geographischen Objekten sowie deren Attribute verändert werden.

Geographische Features können entweder durch Digitalisierung anderer Daten erstellt werden, oder durch Felderhebung mittels mobile mappings beziehungsweise Tracking-Software. Abbildung 11 zeigt die GPS-tracking-Software „MapIt“ für Android. Mit ihr können geographische Objekte im Feld aufgenommen und bearbeitet werden. Möglich ist die Aufnahme von Punkt-, Linien- und Polygongeometrien sowie deren Attribute[19]. Auch der Export in geläufige Formate wie CSV, KML und GeoJSON ist möglich[20], dies vereinfacht den Workflow durch den einfachen Import in ein Desktop-GIS oder als Datengrundlage für ein WebGIS.

[19] MAPITGIS (2107): MapIt User Guide. http://mapit-gis.com/mapit-user-guide/
[20] MAPITGIS (2107): Export Data. http://mapit-gis.com/export-layer-features/

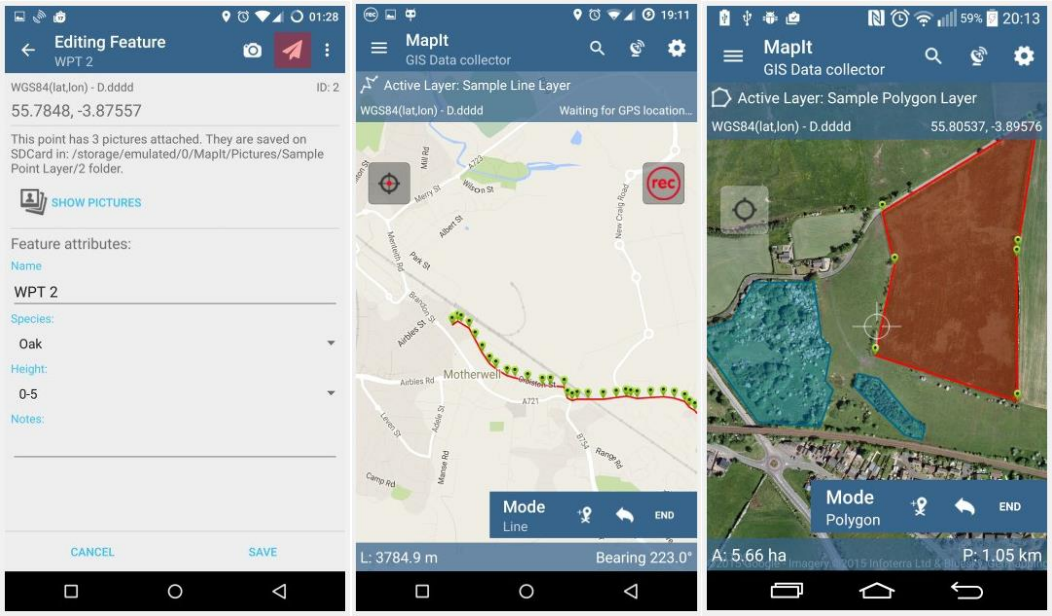


Abbildung 11 GPS-Tracking-Software „MapIt“ zur Erstellung geographische Features im Feld. (Quelle: mapit-gis.com)

3.1.4 CONVERSION

In der Geoinformatik werden trotz vieler Interoperabilitäts- und Standardisierungsbemühungen immer noch häufig Konvertierungstools zwischen verschiedenen Datenformaten benötigt. Die Erstellung und Manipulation von Geodaten werden oft in File- oder Datenbankformaten vollzogen, für die Präsentation oder den Austausch derselben Daten eignen sich meist andere Formate besser.

Man könnte auch argumentieren, dass das Umprojezieren der Geodaten in ein anderes Projektionssystem „Conversion“ darstellt. Die Integrität der Daten sollte sich durch eine solche Transformation nicht verändern, lediglich die Zahlenwerte der Koordinaten sowie der Eintrag des Projektionssystems in den Metadaten werden manipuliert.

Mapshaper ist ein mächtiges freies Web-Tool, unter anderem zur Datenkonvertierung sowie zur Manipulation von Vektordaten durch Generalisierung. Abbildung 12 zeigt eine Generalisierung mittels Douglas-Peuker-Algorithmus, weitere implementierte Algorithmen sind Visvalingam (effective area) sowie Visvalingam (weighted area).[21] Unterstützte Formate sind Shapefile, GeoJSON, TopoJSON, DBF und CSV, der Export ist zusätzlich als SVG möglich. Dies ist besonders für hoch performante statische Webkarten interessant (vgl. Kapitel 2.2.2 beziehungsweise Kapitel 3.3). Mapshaper kann auch als command-line-Tool verwendet werden, sowie in einer eigenen Webapplikation individuell eingebunden werden.

[21] BLOCH, M. (2017): Mapshaper Command Reference. https://github.com/mbloch/mapshaper/wiki/Command-Reference

Abbildung 12 Generalisierung von Vektordaten in Mapshaper, links Ausgangsdaten der Gemeinden um Graz, Rechts die starke Generalisierung mittel Douglas-Peuker-Algorithmus. (Quelle: mapshaper.org)

Ein Web-Tool zur Koordinatentransformation ist epsg.io. Es kann sowohl einzelne Koordinaten direkt in andere Projektionssysteme transferieren sowie Definitionen der Projektionssysteme in verschiedenen Formaten exportieren. Diese sind für server- und clientseitige Transformationen notwendig. Ein weiteres Feature ist die Suche nach Projektionssystemen, entweder über Stichworte oder über die Webkarte.[22]

3.1.5 SHARING

Nach MITCHEL ist auch das Teilen eine Form der geographischen Web-Tools. Dabei bezieht er sich jedoch nicht auf das Teilen der fertigen Kartenvisualisierungen sondern vor allem auf die Verwendung von Webservice-spezifischen Standards zum Teilen der Daten ohne graphisches mapping Interface. Methoden zum Teilen von Geodaten über das Web werden unter anderem durch das OGC spezifiziert, Näheres dazu in den Kapiteln 3.4 und 4.2.

GeoServer ist eine open-source Software zur Veröffentlichung von Geodaten im Web und verwendet OGC-Standards. Daten können serverseitig inklusive Styling bereitgestellt und über verschiedene Datenformate in existierende Webkartentools wie OpenLayers oder Google Maps sowie Desktop-GIS wie ESRI ArcGIS eingebunden werden.[23] Das Styling kann, zum Beispiel in OpenLayers, auch nach dem Datenimport über eigene Styleobjekte oder Stylefunktionen verändert werden.

3.2 Canvas-basierte Webkarten-Tools

In der webbasierten Darstellung von Geodaten müssen häufig sehr große Rasterdaten performant dargestellt werden. Dafür ist Canvas aufgrund seiner Rasterstruktur und Zugriff auf die eigene GPU (graphics processing unit) besser geeignet als eine Vektortechnologie. Da die in Kapitel 2.2.1 erwähnten low-level-Verfahren für fehlerfreies und schnelles Rendering von Rasterdaten äußerst komplex sein können, existieren einige auf diesen Bereich spezialisierte Bibliotheken.

[22] KLOKAN TECHNOLOGIES (2016): EPSG.io: Find Coordinate Systems Worldwide. http://epsg.io/about
[23] OPEN SOURCE GEOSPATIAL FOUNDATION (2014): What is Geoserver? http://geoserver.org/about/

3.2.1 OPENLAYERS

OpenLayers ist eine freie open-source JavaScript-Bibliothek zur Darstellung von Webkarten ohne serverseitige Dependencies. OpenLayers implementiert Industriestandards und OGC-Standards wie WMS, WFS, etc. Es wird von einer Open-Source Community weiterentwickelt und unter der BSD (Berkeley Software Distribution) - License veröffentlicht.[24]

OpenLayers ist zurzeit mit der Version 4.6.5 unter der BSD License verfügbar[25]. Das JavaScript-File mit allen Funktionen hat eine Größe 530 KB. Die Größe kann jedoch durch Bundling und "Tree Shaking" drastisch verkleinert werden, zum Beispiel mit dem module-bundler Rollup[26]. Als „Tree Shaking" bezeichnet man den selektiven Import von Modulen. Dadurch kann eine Datei erstellt werden, welche ausschließlich jene Funktionen enthält, welche wirklich benötigt werden.

3.2.2 LEAFLET

Wie OpenLayers ist auch Leaflet eine open-source JavaScript-Bibliothek zur Darstellung von Webkarten. Der Fokus von Leaflet wird auf Einfachheit, Performanz und Usability gelegt.[27] Die Größe von Leaflet fällt mit 134 KB deutlich geringer aus als bei OpenLayers, allerdings ist der standardmäßige Funktionsumfang geringer. Leaflet vertritt die Idee einer möglichst kleinen Hauptbibliothek, weitere Funktionalitäten können mit 3rd-Party-Plugins der Community hinzugefügt werden.

3.2.3 GOOGLE MAPS JAVASCRIPT API

Die Google Maps JavaScript API wird hier aufgrund ihrer weiten Verbreitung erwähnt. Zu sagen ist, dass das Projekt nicht Open-Source ist und nur eine begrenzte Anzahl von Zugriffen im Monat kostenfrei ist. Dies trifft auch zu, wenn man Google Maps als Basemap für eine OpenLayers oder Leaflet Applikation einbinden möchte. Der Vorteil ist der hohe Wiedererkennungswert der Symbolik der Basemap und die Einsteigerfreundlichkeit der API, der Nachteil ist der teilweise limitierte Funktionsumfang sowie der Verzicht auf Open-Source.

Für die Codebeispiele und die Umsetzung des Praxisteils dieser Masterarbeit wird aufgrund der umfangreichen Funktionalitäten, der großen aktiven Community sowie der geringeren persönlichen Einarbeitungszeit OpenLayers verwendet.

---

[24] OSGEO (2017): OpenLayers Info Sheet. http://www.osgeo.org/openlayers
[25] OPENLAYERS (2017): A high-performance, feature-packed library for all your mapping needs. https://openlayers.org/
[26] SCHAUB, T. (2017): OpenLayers, Rollup, and Closure Compiler. https://bl.ocks.org/tschaub/32a5692bedac5254da24fa3b12072f35
[27] AGAFONKIN, V. (2017): an open-source JavaScript library for mobile-friendly interactive maps. http://leafletjs.com

3.3 SVG-basierte Webkarten-Tools

3.3.1 D3

D3 (Data-Driven Documents) ist, wie bereits in 2.2.2.2 erwähnt, eine JavaScript-Bibliothek zur Manipulation von Webdokumenten über HTML, CSS und besonders SVG. Allerdings kann D3 auch zur Visualisierung von Geodaten verwendet werden. Das Ergebnis ist eine hochperformante, meist statische SVG-Karte. Interaktivität kann über D3 ebenso implementiert werden, zum Beispiel über PopUps oder erweiterte CSS-transitions.[28]

Klassische Interaktionen der Canvas-basierten Webbibliotheken wie Zooming und Panning sind meist nicht möglich, daher sind diese Karten vor allem dann geeignet, wenn die Kartenansicht nicht verändert wird und somit eine Grafik mit wechselnden Daten öfters verwendet werden kann (zum Beispiel Staat mit Bundesländern). Abbildung 13 zeigt eine mit D3 erstellte Karte der US-Präsidentenwahl 2012 der NY Times. Dabei werden Gebiete, in denen im Vergleich zur letzten Wahl eher republikanisch gewählt wurde, als rote, sich nach rechts bewegende Punkte dargestellt, vice-versa die zunehmend demokratischen Gebiete [29]. Hier werden die Veränderungen zwischen den zwei Zeitepochen als Animation, also als intuitiver Fluss in eine Richtung, dargestellt.

[28] BOSTOCK, M. (2017): D3 Data-Driven Documents. https://d3js.org/
[29] THE NEW YORK TIMES COMPANY (2016): Changes in the Electorate in Key States and Regions. http://www.nytimes.com/interactive/2012/11/11/sunday-review/counties-moving.html

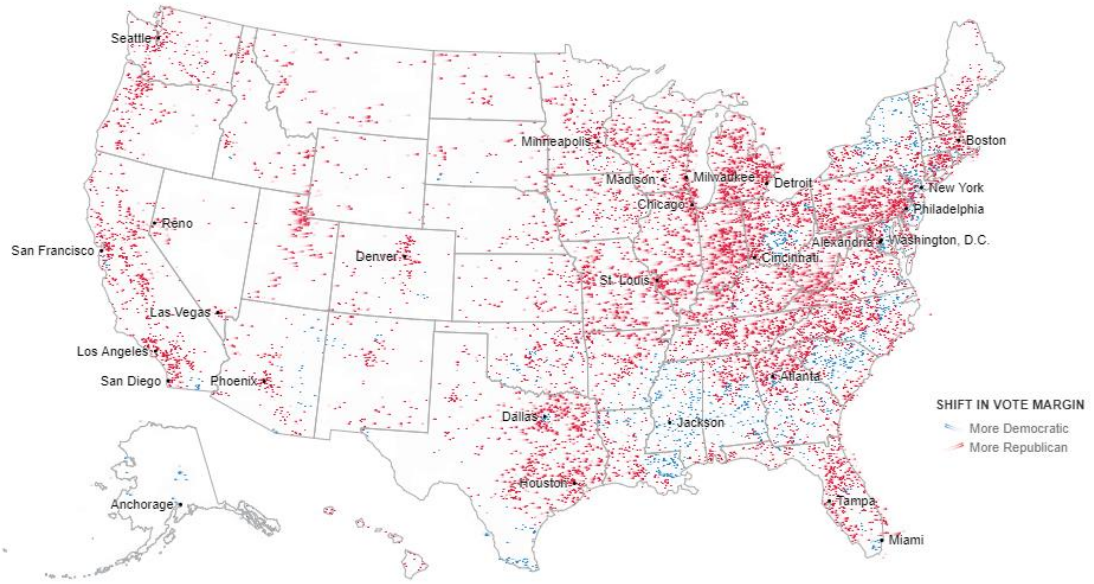


Abbildung 13 D3 Karte mit Animation, der Rechts- und Linksruck der US-Präsidentenwahl 2012 werden durch wortwörtliche Bewegungen der Farbpunkte in die jeweilige Richtung visualisiert. (Quelle: nytimes.com)

D3 kann mit auch mit anderen Bibliotheken zusammen genützt werden. OpenLayers unterstützt D3 nativ, für Leaflet existieren Plugins. Dadurch können mit D3 erstellte Grafiken zusammen mit den Vorteilen der Canvas-basierten Webmaps genutzt werden.

3.4 Basemaps

Basemaps dienen dem Benutzer zur Orientierung. Sie werden im Hintergrund der eigentlichen Karteninhalte dargestellt. Basemaps sind online meist als WMS oder WMTS zugänglich. Alle gängigen WebGIS-Bibliotheken bieten Möglichkeiten die Formate zu unterstützen.

3.4.1 WEB MAP SERVICE (WMS)

WMS ist ein OWS (OGC Web Service) zur Darstellung von Webkarten. In diesem Sinne ist eine Karte die Abbildung von geographischen Daten als Bilddatei, speziell für die Präsentation am Bildschirm. Die Karte wird am Server gerendert, das Bilddatenformat kann variieren. Möglich sind unter anderem PNG, GIF, JPG, SVG, PDF und KML[30].

Die WMS-Spezifikation sieht verschiedene Anfragemöglichkeiten vor[31]:

- GetCapabilities: sendet XML mit Metadaten über den Web Map Server
- GetMap: sendet eine Karte als Bild, erstellt nach den Nutzeranforderungen

[30] OSGEO: WMS output formats. http://geoserver.readthedocs.io/en/latest/services/wms/outputformats.html
[31] MCKENNA, J. (2017): WMS Server. http://mapserver.org/ogc/wms_server.html

- GetFeatureInfo (optional): sendet Information über Daten an einem bestimmten Punkt (zum Beispiel query bei Mausklick)
- DescribeLayer (optional): sendet XML mit Beschreibung eines oder mehrerer Layer
- GetLegendGraphic (optional): sendet eine Legendengraphik eines gewählten Layers als Bilddatei

3.4.2 WEB MAP TILE SERVICE (WMTS)

Der OWS WMTS bedient dieselben Bedürfnisse wie ein WMS, jedoch werden die Bilddaten nicht erst bei der Abfrage mithilfe einer Bounding Box erstellt, sondern bereits im Vorhinein in 256x256 Pixel große Graphiken gerendert. Diese werden gecached, werden sie einmal erstellt, bleiben sie am Server vorhanden und liegen beim nächsten Zugriff bereits fertig bereit.[32] Dadurch steigt die Performance, insbesondere, wenn viele Zugriffe auf den Server zu erwarten sind.

Durch das vorrendern der Basis-Kartendaten zu Bilddateien wird die Performance stark erhöht. Allerdings hat der Benutzer, also der Entwickler der WebApp, keinen Einfluss mehr auf das Styling der Karte.

3.4.3 WEB FEATURE SERVICE (WFS)

Der OGC WFS Standard ermöglicht die Kommunikation mit diskreten geographischen Daten in Vektorform, unabhängig von den Ausgangsdaten. Der Benutzer kann Daten über WFS Daten abfragen, stylen, editieren, und downloaden. Selbst kollaboratives Mapping wird unterstützt.[33]

Da im Zuge dieser Masterarbeit die Basemap lediglich als Hintergrund zur Orientierung dienen soll, wären die meisten Funktionen des WFS nicht notwendig. Die Möglichkeit des Stylings der Daten könnte nützlich sein, zum Beispiel damit sich die Grundkarte farblich an die dargestellten Daten des Benutzers anpassen kann, sollte der umgekehrte Weg nicht wünschenswert sein.

3.4.4 WEB COVERAGE SERVICE (WCS)

Der OGC Web Coverage Service ist spezialisiert auf weitgehende Datenkommunikation mit Raum/Zeit-abhängigen Rasterdaten. WCS ermöglicht den Zugriff auf Datenstrukturen, welche auch das clientseitige Rendering sowie die Integration der Daten in Modelle und Analysen ermöglicht. Im Unterschied zu WFS, welches diskrete Werte liefert, können mit WCS multidimensionale geotemporale

[32] ESRI (2017): WMTS Services. http://server.arcgis.com/en/server/latest/publish-services/linux/wmts-services.htm
[33] OSGEO (2017): WFS reference. http://geoserver.readthedocs.io/en/latest/services/wfs/reference.html

Werteflächen kommuniziert werden. WCS 2.0 benutzt den Rahmen des OGC GML Application Schema. (ABDALLA 2016, S. 130)

WCS wird oft als Raster-Pendant zu WFS gehandelt. Es ist besonders nützlich für kontinuierliche Rasterdaten, wie Höhenmodelle oder Temperaturen. Dadurch, dass sowohl die Zeit als auch die Ausdehnung der Daten variabel sein können, eignet sich dieses Format auch für Katastrophenmanagement wie Unwetter, Überflutungen oder radioaktive Unfälle. WCS ist ein relativ junger Standard mit geringer Grunddatenverfügbarkeit. Die Unterstützung der Web Mapping Libraries ist demnach noch eingeschränkt.

3.4.5 FREI VERFÜGBARE BASEMAPS

Die meisten online verfügbaren Basemaps basieren entweder auf den Daten der Open Street Map oder auf staatlichen Datensätzen. Hier werden einige Basemaps vorgestellt sowie auf Verfügbarkeit und Anwendungsgebiete geprüft.

3.4.5.1 Mapnik

Mapnik ist keine Basemap an sich, sondern eine open-source Software, mit der eigene Grundkarten erstellt werden können. Die meisten online erhältlichen Basemaps basieren auf OSM-Daten und wurden mit Mapnik gestyled und gerendert. Mapnik ist verwendbar mit C++, Python sowie Node. Stylesheets, die das Aussehen der Karte bestimmen, werden im XML-Format erstellt.[34]

Code 3 Beispielhaftes Mapnik-Stylesheet im XML-Format (Quelle: mapnik.org)

```
<Map background-color="blue" srs="+init=epsg:4326">
  <Style name="My Style">
    <Rule>
      <PolygonSymbolizer fill="#f2eff9" />
      <LineSymbolizer stroke="rgb(50%,50%,50%)" stroke-width="0.1" />
    </Rule>
  </Style>
  <Layer name="world" srs="+init=epsg:4326">
    <StyleName>My Style</StyleName>
    <Datasource>
      <Parameter name="file">path/to/shapefile.shp</Parameter>
      <Parameter name="type">shape</Parameter>
    </Datasource>
  </Layer>
</Map>
```

Die Möglichkeit, allgemein gültige Regeln für die Visualisierung von Geodaten zu erstellen und damit seine eigene Grundkarte automatisiert zu erzeugen, kann für einige Fragestellungen äußerst nützlich sein. Insbesondere dann, wenn die Webkarte für die ganze Welt auch in sehr hohen Zoomstufen verfügbar sein soll und die bereits existierenden Grundkarten aus verschiedenen Gründen nicht akzeptabel sind. Dieser Fall kann zum Beispiel bei der Überlagerung eines oder mehrer Layer eintreten,

[34] PAVLENKO, A. (2017): mapnik. http://mapnik.org/

sodass die Karte zu unübersichtlich werden würde. Oder man benötigt nur einzelne Objekttypen, wie etwa das Verkehrsnetz, dieses jedoch global, hochperformant und mit einer speziellen Farbgebung. Durch den sehr hohen Aufwand, seine eigenen Visualisierungsregeln zu erstellen und durch das recht hohe Angebot an frei verfügbaren Basemaps, ist diese Methode nur in Ausnahmefällen zu empfehlen.

3.4.5.2 Open Street Map

Open Street Map bietet nicht nur die offenen Geodaten, sondern auch eine über Mapnik gerenderte Standardkarte (OpenStreetMap Carto oder osm-carto) als Basemap. Das Design dient als Allzweckkarte in allen Zoomstufen. So werden viele Spezialinformationen der Open Street Map wie Bahnstreckendetails oder Schifffahrtszeichen nicht abgebildet. Die Labels sind in der jeweiligen Landessprache. Die Mapnik-Stylesheets wurden mit der Software CartoCSS vorprozessiert und sind, wie das gesamte OSM-Projekt, Open Source. [35]

Das Projekt wurde mehrfach als Basis für weitere Kartenansichten benutzt. Die „OSM Bright“ bedient dieselben Bedürfnisse wie die OSM Carto, wirkt jedoch viel aufgeräumter. Besonders die Landbedeckung rückt hier optisch in den Hintergrund (siehe Abbildung 14). Eine deutsche Version, die optisch den in Deutschland weit verbreiteten und somit vertrauten „Schell-Atlas“ nachempfunden wurde (siehe Abbildung 15), ist vor allem für jene Benutzer gedacht, die jahrelang mit genannten Karten gearbeitet haben.[36] Dadurch werden Kartenelemente sowie Zeichen intuitiver erkannt. Auch eine französische und eine schweizer Basemap wurden nach demselben Prinzip erstellt.

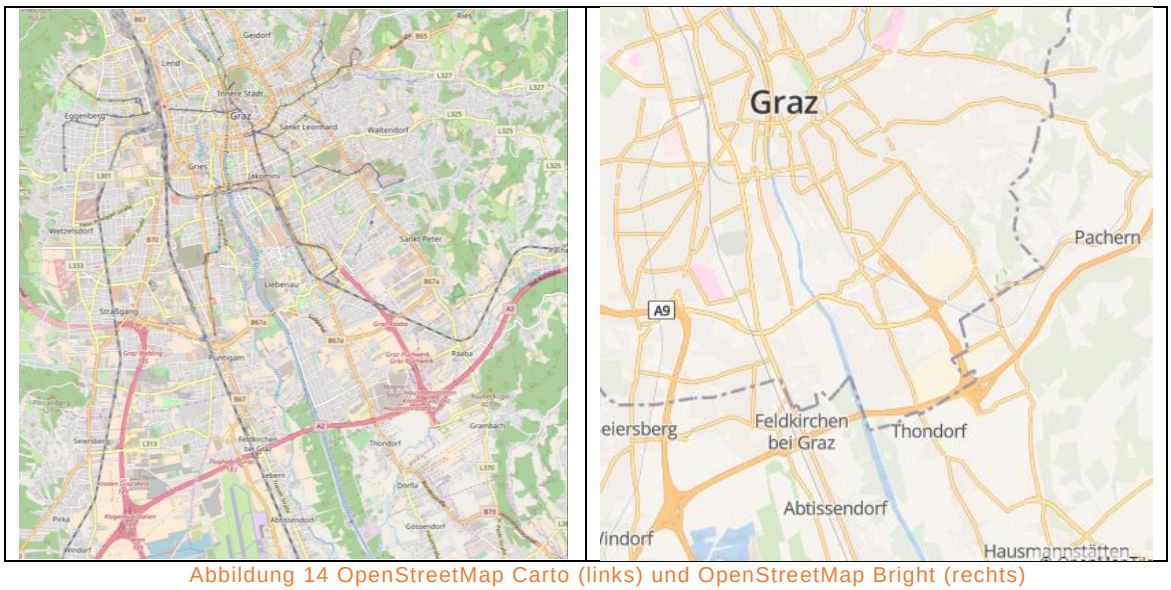


Abbildung 14 OpenStreetMap Carto (links) und OpenStreetMap Bright (rechts)
(Quellen: openstreetmap.org beziehungsweise openmaptiles.org)

[35] OSM: Standard tile layer. http://wiki.openstreetmap.org/wiki/Standard_tile_layer
[36] OSM: German Style. http://wiki.openstreetmap.org/wiki/German_Style

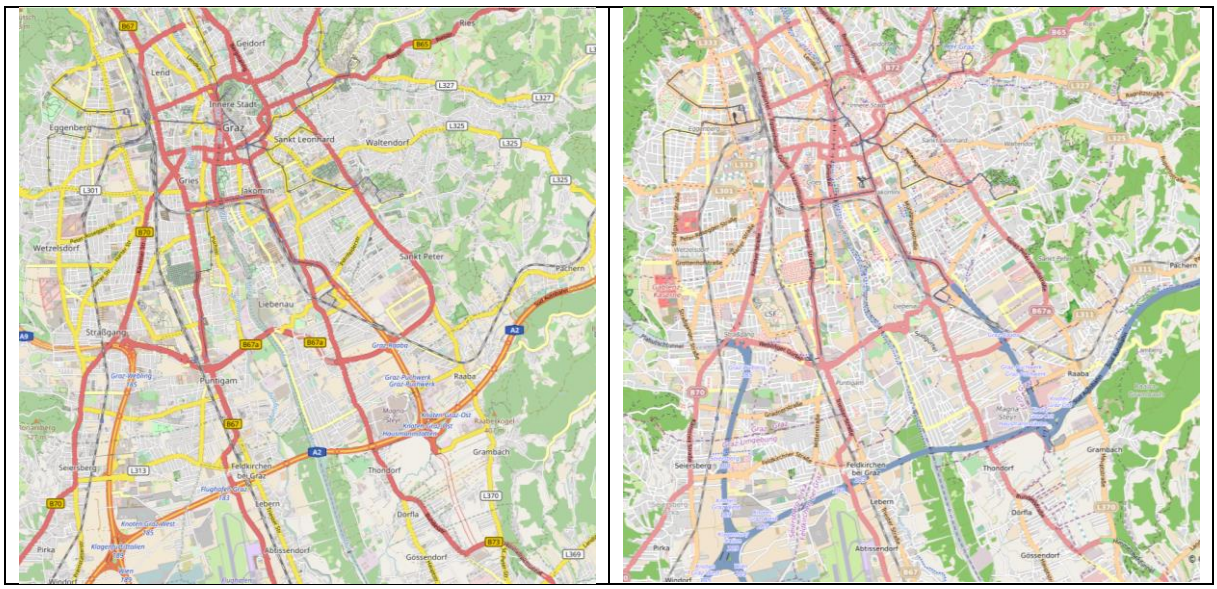


Abbildung 15 OS deutscher Stil (links) und französischer Stil (rechts) (Quellen: openstreetmap.de beziehungsweise openstreemap.fr)

3.4.5.3 Stamen Maps[37]

Stamen Design ist ein Entwicklerstudio spezialisiert auf Datenvisualisierung mit besonderem Fokus auf humanitäre Arbeit. „Stamen Maps“ ist eines ihrer vielen frei verfügbaren Projekte. Ihre Hauptkarten sind „Toner“, „Terrain“, und „Watercolor“, wobei jede dieser Karten in verschiedenen „Flavors“, also leichten Abweichungen, verfügbar sind.

- Toner
  Toner ist eine radikal monochrome Basemap dediziert für „Data Mashups“, also Karten, in denen andere Daten als Overlays (zum Beispiel als Diagramme) dargestellt werden. Auch Küstenzonen und Flußverläufe sind besonders klar sichtbar. Für Toner sind unter anderem die Versionen „Standard Toner“, „Lines“ und „Lite“ verfügbar (siehe Abbildung 16). Durch die monochrome Darstellung der Basemaps hat der Entwickler einer „Mashup“-Karte mehr Freiheiten bezüglich Farbgebung.

[37] STAMEN: http://maps.stamen.com/#toner/12/37.7406/-122.3058

Abbildung 16 Stamen Maps: Toner. Links: Standard Toner, mitte: Lines, rechts: Lite. (Quelle: maps.stamen.com)

- Terrain

  Die Terrainkarte bietet ein Hillshade und natürliche Höhenfarben. Besonderer Fokus wird hier auf die Platzierung der Labels sowie der Generalisierung mehrspuriger Straßen gelegt. Die Stamen Terrain Map ist unter anderem in den Flavours „Standard Terrain“, „Lines“ und „Background“ erhältlich (siehe Abbildung 17).

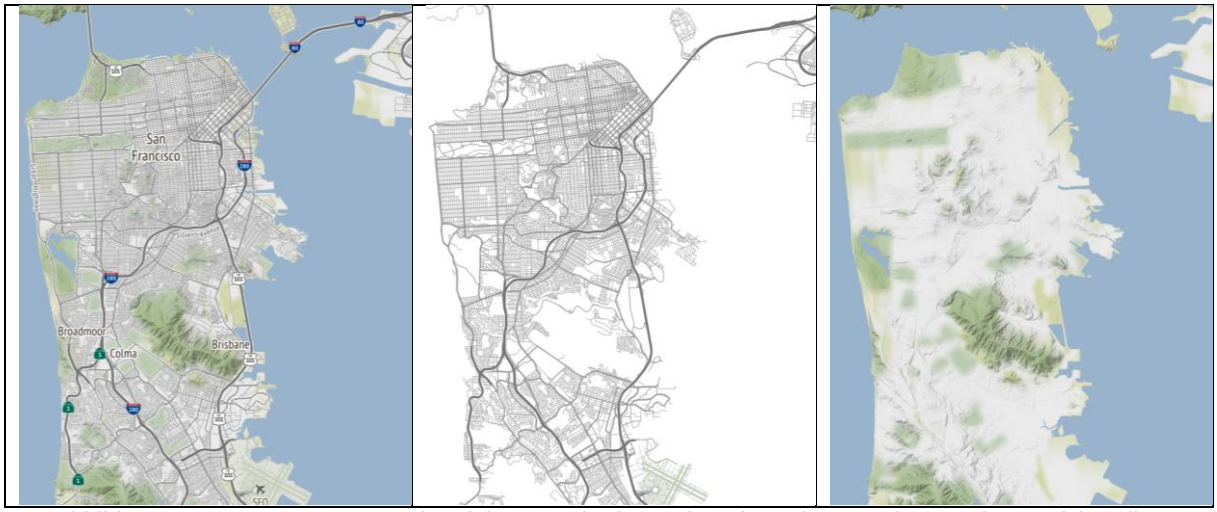


Abbildung 17 Stamen Maps: Terrain. Links: Standard Terrain, mitte: Lines, rechts: Background (Quelle: maps.stamen.com)

- Watercolor

  Die Watercolor-Karte soll an ein handgemachtes Wasserfarbengemälde erinnern. Die Karte ist sehr bunt und sorgt besonders im Web durch die Verschneidung von traditionellem Design und der Interaktivität der Webkarte für Aufmerksamkeit beim Betrachter (siehe Abbildung 18). Für Datenmashups ist sie weniger geeignet, allerdings als Eye Catcher, sollte diese Eigenschaft notwendig sein. Standardmäßig sind keine Labels vorhanden, jedoch bieten die oben genannten Karten eigene Label-flavours, welche auch über diese Karte gelegt werden können.

Abbildung 18 Stamen Watercolor (Quelle: maps.stamen.com)

3.4.5.4 basemap.at

Basemap.at ist ein Projekt der neun österreichischen Bundesländer, ITS Vienna Region, TU Wien und SynerGIS. Ihre Basis sind Geodaten der österreichischen Verwaltung, somit ist für die Datenerhebung keine freiwillige Community notwendig. Wie die OSM sind auch die Grundkarten von basemap.at selbst für kommerzielle Anwendungen ohne Zugriffslimitierungen frei verfügbar. Die Grundkarten sind unter anderem in den Varianten Standard, Grau und Orthofoto erhältlich (siehe Abbildung 19), sowie einer hochauflösende High-DPI (Dots per Inch) Version der Standardkarte.[38]

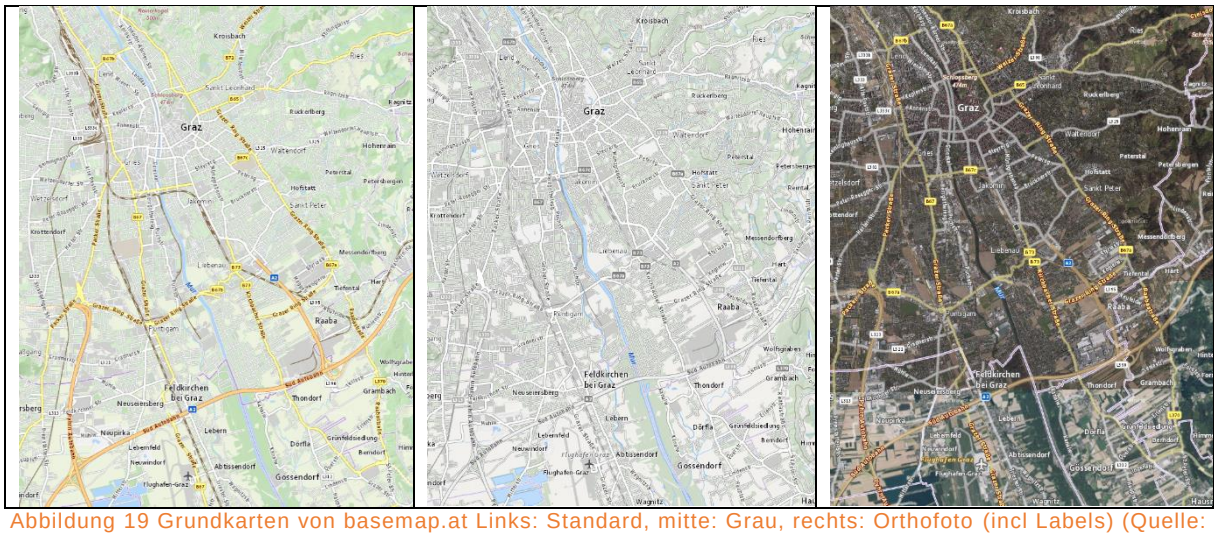


Abbildung 19 Grundkarten von basemap.at Links: Standard, mitte: Grau, rechts: Orthofoto (incl Labels) (Quelle: basemap.at)

Dadurch, dass basemap.at das gesamte österreichische Staatsgebiet abdeckt, ist es nur für Kartenapplikationen wie Datenmashups für Österreich geeignet. Eine Überlagerung mit anderen Grundkarten wäre zwar technisch gesehen möglich, ist aufgrund der unterschiedlichen Kartographie jedoch nicht ratsam.

[38] STADT WIEN et al (2016): Willkommen bei basemap.at – der Verwaltungsgrundkarte von Österreich. https://basemap.at/

3.5 Map Design im Web

Map Design umfasst zwei unterschiedliche Thematiken: zum einen das Layout von Designelementen, zum anderen das Planen der Karte. Das Layout behandelt Fragen wie „Wo soll der Titel platziert werden, wo die Legende und der Maßstab?“. In der bildenden Kunst nennt man dies Komposition. Design im Sinne der Planung beginnt vor dem ersten Strich und beinhaltet welche Informationen eingebunden werden, welche Projektion, welcher Maßstab und welche Art von Symbolen verwendet werden. Sie stellt das Herz der Kartenerstellung dar. (TYNER 2010, S. 18)

TYNER erkennt einige Grundziele des gestalterischen Mapdesigns (nach TYNER 2010, S. 19 ff). Sie sind sowohl für klassische Karten im Printbereich sowie für moderne interaktive Karten im Web anwendbar.

3.5.1 KLARHEIT

Klarheit beinhaltet das Prüfen der Karte auf etwaige Störfaktoren. Eine klare Karte bringt die wichtigen Aspekte zur Geltung. Jedes weitere gestalterische Objekt, welches nicht dem eigentlichen Zweck der Karte dienlich ist, ist nur ein Störfaktor und verbessert somit nicht die Botschaft der Karte.

3.5.2 ORDNUNG

Ordnung bezieht sich auf die Logik der Karte. Erzeugt die Karte Verwirrung oder ein visuelles Durcheinander? Sind die Kartenelemente logisch platziert beziehungsweise dort wo ein Benutzer sie erwarten würde? Wird das Auge des Benutzers adäquat durch die Karte und die Kontrollelemente geführt?

Eine Karte ist ein synoptisches Kommunikationsmedium, der Kartograph kann nicht von einer klaren Betrachtungsreihenfolge (wie etwa zuerst Titel, dann Legende, dann Karteninhalt) ausgehen. Studien der Augenbewegungen beim Kartenlesen zeigen beträchtliche Bewegungen zwischen einzelnen Elementen auf einem Kartenblatt beziehungsweise auf einem Bildschirm. Es ist jedoch bekannt, dass mit Hilfe von gewissen geometrischen Formen und Darstellungskonventionen der Blick des Betrachters „geführt“ werden kann.

Anzumerken ist hier, dass bei automatisiert und semiautomatisiert erstellten Karten auch der Karteninhalt an sich die Ordnung beeinflussen kann. Vorausgesetzt die Karten sind genordet, hat eine Karte des Staates Chiles durch die Form des abgebildeten Objektes einen anderen Einfluss auf die Augenwege als eine Karte Russlands. Eine thematische Karte mit sehr dichten Messpunkten hat eine andere Ordnung als eine Karte mit nur wenigen Messpunkten. Die Ausgangsdaten sind jedoch in diesen Fällen nicht beeinflussbar, es kann nur das restliche Kartendesign angepasst werden.

3.5.3 BALANCE

Jedes graphische Element hat eine visuelle Höhe. Diese Gewichte sollten gleichmäßig um das optische Zentrum der Seite verteilt sein, welches etwas über dem geometrischen Zentrum der Seite liegt. Andernfalls wirkt die Karte einseitig oder instabil.

Ein weiterer Aspekt der Balance ist der “White Space”. Dabei handelt es sich um gestalterischen Leerraum, um die Karte aufzulockern und das Auge nicht zu überfordern. Grundsätzlich sollte das Kartenbild besonders bei Bildschirmkarten so groß wie möglich sein, White Space ist vor allem bei der Gestaltung äußerer Elemente wichtig.

Bei Karten kann grundsätzlich davon ausgegangen werden, dass der Betrachter von sich aus die Karte lesen möchte und nicht erst darauf aufmerksam gemacht werden muss. Die von TYNER beschriebene “Balance“ kann jedoch auch über graphische Gestaltung absichtlich aus dem visuellen Gleichgewicht gebracht werden, sollte es notwendig sein, die Aufmerksamkeit auf sich zu ziehen.

3.5.4 KONTRAST

Ein großer Teil der Klarheit der Karte ist abhängig vom Kontrast, also dem optischen Unterschied zwischen den einzelnen Elementen. Kontrast bezeichnet optische Unterschiede wie hell und dunkel, dick und dünn, leicht und schwer. Mit Kontrast kann eine Art „Tiefe“ suggeriert werden, die für den Kartenzweck wichtigen Elemente sollen sich klar vom Hintergrund abheben. Ist eine Webkarte interaktiv und weist klickbare Schaltflächen auf, so müssen diese sich wiederum klar von den Kartenelementen abheben.

3.5.5 EINHEIT

Einheit bezieht sich nach TYNER auf die Wechselbeziehung zwischen Kartenelementen. Elemente wie Kartenschriften müssen untereinander abgestimmt, Piktogramme nach einem einheitlichen Darstellungsschema erstellt werden. Der Vordergrund muss mit dem Hintergrund in Einklang gebracht werden, Symbole dürfen nicht untereinander im Konflikt stehen und müssen an das Kartenthema angepasst sein.

Gerade bei Webkarten sind Basemaps üblich (siehe Kapitel 3.4: *Basemaps*). Diese werden von Webkartographen unter Berücksichtigung all dieser Faktoren erstellt, können aber nicht ohne Weiteres verändert werden. Das bedeutet, wenn eine vorgefertigte Basemap einen substanziellen Teil des Kartenbildes darstellt, so müssen weitere Kartenelemente an diese angepasst werden.

Oft wird eine Karte auch als kleine Grafik in ein anderes Medium eingebettet, zum Beispiel in eine Zeitung oder in eine Webseite. In diesem Fall kann es notwendig sein,

auch das bereits bestehende Design des umgebenden Mediums in die Karte einzubinden.

3.5.6 HARMONIE

Unter Harmonie versteht man das optische Zusammenspiel aller Elemente. Vertragen sich die Farben untereinander oder wird Spannung erzeugt? Funktionieren die graphischen Elemente miteinander? Erzeugt die Karte ein angenehmes Erscheinungsbild?

Wie bereits in Kapitel 3.5.3 erwähnt, kann auch die Harmonie bewusst etwas aus dem Lot gebracht werden. Durch das Kreieren von optischer Spannung kann, falls notwendig, Aufmerksamkeit erzeugt werden. Grundsätzlich werden jedoch harmonische Karten bevorzugt, die Harmonie kann den Unterschied zwischen Akzeptanz und Ablehnung einer Karte ausmachen.

3.6 Interaktionen

Interaktionen des Benutzers sind eines der größten Vorteile von Webapplikationen gegenüber Printmedien. Klassische Webkarteninteraktionen wie Zooming mit dem Mausrad und Panning mit gehaltener Maustaste werden bei vielen Anwendungen bereits erwartet. Werden Features visualisiert, so ist eine Anzeige weiterer Informationen zu einem durch Einzelklick gewählten Feature naheliegend. Diese Annahme kann, wenn die Features tatsächlich geklickt werden sollen, durch Hover-Effekte verstärkt werden.

Weitere Benutzerinteraktionen müssen entsprechend kommuniziert werden. Möglich sind unter anderem:

- Upload von Daten
- Filterung der Daten
- Analyse der Daten
- Veränderung der Visualisierungsparameter
- Selektion eines einzelnen Features
- Selektion mehrerer Features

Jede Interaktion hat Berechnungen und somit response Times zur Folge. Nach KALAWSKY (2009, S. 129) können response Times folgendermaßen gegliedert werden:

Tabelle 3 Die Auswirkung der response Time auf eine Benutzerinteraktion (nach KALAWSKY 2009, S. 129)

0.1s	Dies ist ungefähr das Limit, bei dem der Benutzer das Gefühl einer sofortigen Reaktion des Systems erhält. Unter normalen Umständen ist kein weiteres Feedback an den Benutzer notwendig, er sieht die Ergebnisse der Interaktion ohne fühlbaren beziehungsweise relevanten Verzug.
1s	Bis zu einer Sekunde bleibt der Gedankenablauf des Benutzers ungestört. Allerdings verspürt der Benutzer bei dieser response Time häufig einen Kontaktverlust zu den Daten. Bei einer response Time zwischen 0.1 und 1s sollte ein Feedback an den Benutzer zurückgegeben werden.
10s	Dies stellt das Limit der Aufmerksamkeit des Benutzers auf das Interface dar. Bei längeren response Times wird der Benutzer abgelenkt und wird oft versuchen, andere Aufgaben auszuführen bis die ursprüngliche Aufgabe abgeschlossen ist. Im Idealfall sollte dem Benutzer ein Feedback in Form eines Fortschrittindikators geboten werden (siehe Abbildung 20). Ist die response Time des Systems nicht vorhersehbar, ist ein solcher Indikator sogar noch wichtiger.

Abbildung 20 zeigt eine Möglichkeit eines unaufdringlichen “Progress Indicators“ in OpenLayers. In diesem Fall wird nicht der Fortschritt der Datenanalyse oder der Datenvisualisierung überwacht, sondern das Laden der einzelnen Kacheln einer Basemap. Dies wird durch das Feuern des Events `tileloadstart` für jede Kachel beim Laden eines Ausschnitts eines WMTS-Layers (`ol.source.WMTS`) ermöglicht. Bei Vollendigung des Ladevorgangs feuert jede Kachel das Event `tileloadend`.[39]

Der Balken an sich ist lediglich ein leeres HTML `<div>`-Element mit blauer Hintergrundfarbe, einer Höhe von 2px und einer Breite von 0% der Breite der Karte. Wird in einen Bereich gezoomt, dessen Tiles noch nicht geladen wurden, stellt die Anzahl der gefeuerten `tileloadstart`-Events im Verhältnis zu den bereits vollendeten `tileloadend`-Events den aktuellen Ladefortschritt dar. Dieser kann als Prozentangabe über das CSS-Attribut `width` als Breite des Ladebalkens wiedergegeben werden.[40] Es kann passieren, dass einige Tiles bereits fertig geladen wurden, bevor der Ladevorgang aller Tiles begonnen hat. Eine exakte Wiedergabe des Ladevorganges ist jedoch gar nicht notwendig, da dem Benutzer nur kommuniziert werden soll, dass eine Aktion im Gange ist, und er geduldig bleiben muss.

[39] OPENLAYERS (2017): ol.source.WMTS https://openlayers.org/en/latest/apidoc/ol.source.WMTS.html
[40] OPENLAYERS (2017): Tile Load Events. https://openlayers.org/en/latest/examples/tile-load-events.html

Die Fortschritte einer Analyse oder Visualisierung können je nach Anwendung und Notwendigkeit entweder in denselben Balken einbezogen werden, eine zweite Grafik parallel dazu darstellen oder den Balken komplett ersetzen. Weitere übliche Darstellungsformen von Progress Indicators wären radiale Animationen (Endlosschleife als “Spinners“ oder deterministisch bis 100%), oder sich bewegende Zahnräder[41].

Abbildung 20 Darstellung des Fortschrittes des Ladens der Basemap durch einen Progress Bar in Openlayers als blauer Balken im Bild unten. (Quelle: openlayers.org)

Bei Interaktionen mit Webkarten ist manchmal die Erzeugung einer Illusion einer schnellen response Time möglich. Besonders bei Zoom-Interaktionen oder nach einer geographischen Filterung der Daten und anschließendem Zooming oder Panning zu den übrig gebliebenen Features ist es möglich, diese Animationen zumindest mit der Basemap durchzuführen, während die Berechnung der Visualisierung der eigentlichen Daten noch im Gange ist.

Auch die clientseitige Analyse der Daten kann in vielen Fällen kürzere response Times liefern als eine serverseitige Analyse, da die Serverkommunikation inklusive Datentransfer entfällt. Handelt es sich jedoch um große Datenmengen (“Big Data“) oder sehr komplexe Analysen, so können serverseitige Analysen durch die größere Rechenleistung bessere Ergebnisse erzielen.

3.7 Diskurs: Farbenlehre

Die Benutzung von Farbe in Diagrammen und anderen graphischen Darstellungen kann den Benutzer gleichermaßen unterstützen wie verwirren. In diesem kurzen Diskurs wird auf die Farbenlehre in der Diagrammdarstellung eingegangen sowie auf das Zusammenspiel von Farben in Diagrammen und Karten.

Der großzügige Einsatz von Farbe ist verlockend für die graphische Darstellung von Daten. Allerdings muss darauf geachtet werden, wie Farbe verwendet wird, ansonsten kann sich Farbe unnötigerweise negativ auf die Lesbarkeit der Graphik auswirken.

[41] BABICH, N. (2016): Best Practices For Animated Progress Indicators https://www.smashingmagazine.com/2016/12/best-practices-for-animated-progress-indicators/

Es existiert eine große Anzahl verschiedener Möglichkeiten, Farben zu beschreiben. Herkömmliche Computerbildschirme verwenden additive Farbmischung, welche durch das Mischen von Rot, Grün und Blau die Farbe Weiß entstehen lässt. Der RGB-Farbraum beschreibt Farben aus seinen Rot- Grün- und Blauwerten, welche größtenteils Werte von 0 bis 100 oder von 0 bis 255 annehmen können. Eine weitere Möglichkeit, Farben im Web zu beschreiben ist über die subtraktive Farbmischung durch HSL (Hue, Saturation, Lumination) beziehungsweise HSB (Hue, Saturation, Brightness). Diese beschreibt eine Farbe bestehend aus dem Farbton, der Sättigung und der Helligkeit. Für den Farbton ist ein Farbkreis definiert, der bei Rot beginnt und sich über Gelb, Grün, Blau und Violett schließt. Der Wertebereicht des Farbtons liegt bei 0 bis 360, die der Sättigung und Helligkeit jeweils bei 0% bis 100%.[42]

Um quantitative Unterschiede effizient darzustellen, sollte eine Abstufung von Grautönen verwendet werden. Die meisten Kartenbenutzer sind in der Lage, fünf bis sechs Helligkeitskontraste zu unterscheiden, wobei die Unterscheidbarkeit von dunkleren Helligkeiten leichter ist als die von helleren. Farbtöne sind für die Darstellung von Quantitäten ungeeignet. Im Unterschied zu Grautönen können diese nicht einfach logisch geordnet werden. Hinzu kommen häufige Beeinträchtigungen wie Rot-Grün-Schwäche. Eine Legende kann zwar hilfreich für die Lesbarkeit einer solchen Karte sein, allerdings niemals effizient. (MONMONIER 1991, S. 21ff)

FEW (2012, S. 72) erstellte zu diesem Thema eine Tabelle aller graphischen Variablen und bewertete sie nach ihrer Qualität, quantitative Eigenschaften vorbewusst (präattentiv) zu übermitteln:

Tabelle 4 Präattentive Eigenschaften visueller Variablen bezüglich quantitativer Wahrnehmung (Quelle: FEW 2012, S. 72)

Typ	Attribut	Quantitativ wahrnehmbar?
Form	Länge	Ja
	Breite	Ja, aber limitiert
	Orientation	Nein
	Größe	Ja, aber limitiert
	Form	Nein
	Einschluss	Nein
Farbe	Farbton	Nein
	Intensität	Ja, aber limitiert
Position	2-D Position	Ja

Auch nach FEW kann der Farbton vorbewusst keine quantitative Information enthalten. Demnach ist eine Legende notwendig beziehungsweise muss der Betrachter die Darstellung „lernen“, um Werte zu extrahieren. In der App Mapoch wird Hue nicht in

[42] BROADLEY, C. (2018): How do colors work on the web in 2018? https://digital.com/blog/web-colors/

erster Linie als graphische Variable für absolute quantitative Werte benutzt, es soll lediglich eine grobe ordinale Einordnung möglich sein (ein Wert ist größer/besser als ein anderer).

Zu beachten ist in Tabelle 4, dass in Karten beziehungsweise in Kartogrammen die 2D Position meist bereits für die geographische Position der Objekte reserviert ist. Auch die graphischen Variablen der Länge und der Breite sind zwar theoretisch zur Darstellung von Quantitäten angemessen, allerdings ist deren quantitative Wahrnehmbarkeit eingeschränkt, da die gemeinsame Basislinie der Objekte wie in einem Diagramm wegfällt.

Die Verwendung unterschiedlicher Farbtöne ist sinnvoll zur wertfreien Differenzierung verschiedener Attribute in Diagrammen und ähnlichen Grafiken. FEW hat zu diesem Zwecke drei harmonische Farbpaletten entwickelt, welche eine solche wertfreie Differenzierung ermöglichen sollen (nach Few 2012, S. 344):

Tabelle 5 Light Palette: Für Datenkodierung in große Objekten wie Balken oder Boxen (Quelle: Few 2012, S. 344)

	Rot	Grün	Blau
	140	140	140
	136	189	230
	251	178	88
	144	205	151
	246	170	201
	191	165	84
	188	153	199
	237	221	70
	240	126	110

Tabelle 6 Medium Palette: Für Datencodierung in kleine Objekte wie Punkte oder Linien (Quelle: Few 2012, S. 344)

	Rot	Grün	Blau
	77	77	77
	93	165	218
	250	164	58
	96	189	104
	241	124	176
	178	145	47
	178	118	178
	222	207	63
	241	88	84

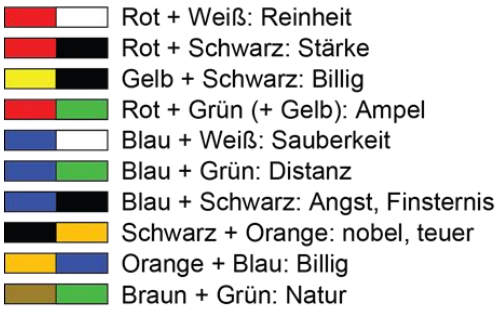
Tabelle 7 Dark&bright Palette: Zum Hervorheben eines bestimmten Gegenstandes, wie etwa eines bestimmten Balkens (Quelle: Few 2012, S. 344)

	Rot	Grün	Blau
	0	0	0
	38	93	171
	223	92	36
	5	151	72
	229	18	111
	157	114	42
	123	58	150
	199	180	46
	203	32	39

Jeder dieser Farbtöne besitzt ein Mindestmaß an Unterscheidbarkeit, auch die Reihenfolge der Farben ist nicht zufällig, sondern sorgt für ausreichend Kontrast in der gewählten Anordnung. Je nach Kontext können jedoch einzelne Farben dieser Liste verworfen werden. FEW nennt in etwa kulturelle Unterschiede, zum Beispiel wird Rot in den westlichen Kulturen häufig als Warnfarbe eingesetzt, in China wiederum ist Rot oft ein Zeichen für Glück. (FEW 2012, S. 77)

Im Unterschied zu kulturell gelernten Assoziationen von Farben ist deren psychologische Wirkung von der persönlichen Lebenserfahrung abhängig und variiert somit leicht von Person zu Person. Damit diese Wirkung einer Farbe oder Farbkombination zu tragen kommt, muss es sich um eine große farbige Fläche handeln. Kleine Objekte besitzen so gut wie keine Farbwirkung, können aber Assoziationen herbeirufen. (KORTHAUS 2013, S. 139ff) Die Farbwirkung sollte deshalb besonders bei großen Layoutflächen beachtet werden. In der Literatur werden die psychologischen Wirkungen folgendermaßen beschrieben (KORTHAUS 2013, S. 147-165 und FRIES & WITT 2007, S. 99):

- Gelb: warm, heiter, leuchtend, dynamisch
- Rot: warm, heiß, liebend, aggressiv, mächtig, tatkräftig
- Blau: kalt, fern, tief, unendlich, beruhigend, passiv, entspannend
- Grün: angenehm, unaufdringlich, wachsend, realistisch, faul
- Orange: warm, positiv, stimulierend, fröhlich, lustig, laut
- Violett: mystisch, gelassen, finster, einsam, statisch
- Braun: gemütlich, unrein, faul, dreckig, spießbürgerlich, zurückgezogen
- Schwarz: dunkel, finster, elegant, pessimistisch
- Weiß: hell, rein, leicht, unschuldig, realitätsfern
- Grau: gleichgültig, unbeteiligt



In der Praxis ist besonders für Kartographen auch der graphische Kontext wichtig. Farben wirken anders in der Nähe anderer Farben. In dynamischen Webkarten existiert eine Vielzahl an möglichen, oft ungewollten, grafischen Einflüssen durch die Hintergrundkarten. Um diese unvorhersehbaren Störungen zu minimieren, sollten bei Karten, in denen Overlay-Graphiken wie Diagramme im Vordergrund stehen sollen, möglichst dezente Hintergrundkarten mit niedrigem Kontrast und geringer graphischer Varianz verwendet werden. Außerdem können Rahmen (Borders) oder Hintergründe bei den Diagrammoverlays helfen, den notwendigen Kontrast zum Hintergrund herzustellen.

4 GEOTEMPORALE DATEN

Unter „geotemporalen“ Daten versteht man Daten mit einem Raum- und Zeitbezug. Zeit kann meistens nicht direkt wahrgenommen werden, sondern sie äußert sich durch Veränderungen, die mit dem Verlauf der Zeit einhergehen. Nach KRAAK (2008, S. 294) können diese Veränderungen entweder von temporaler Seite, oder von räumlicher Seite auftreten. Als Veränderungen im räumlichen Bereich nennt er:

- Erscheinen und Verschwinden
- Bewegung (zum Beispiel der Position/Lage, Veränderung der Geometrie/Form)
- Mutation (Veränderung der Eigenschaft eines Phänomens, Zunahme/Abnahme eines Wertes)

Als Veränderungen im temporalen Bereich nennt er:

- Zeitpunkt, in dem eine Veränderung auftritt
- Geschwindigkeit, mit der eine Veränderung über Zeit auftritt
- Reihenfolge, in dem Veränderungen auftreten
- Dauer, in der Veränderung auftritt
- Häufigkeit, in der sich eine Phase wiederholt

Dieses Kapitel befasst sich mit den Eigenheiten von geotemporalen Daten sowie dem Status Quo von deren Visualisierung.

4.1 Geometrien mit Zeitbezug

4.1.1 PUNKTE

Viele erhobene Geodaten können als Punktgeometrien repräsentiert werden. Die meisten Messgeräte messen nur an einem Punkt, auch GNSS-Positionen sind Punkte. Hat man mehr als einen Messpunkt, so möchte man diese zur leichteren Vergleichbarkeit oft in einer Karte darstellen.

Es existieren viele Möglichkeiten, Punktdaten und ihre Attribute in Karten darzustellen. Die richtige Darstellungsart leitet sich zum einen vom Verwendungszweck, zum anderen von den Daten selbst ab. Hier werden die Darstellungsmöglichkeiten behandelt, wenn jeder einzelne Messpunkt dargestellt werden soll. Manche Messdaten können auch auf Linien- oder Polygongeometrien aggregiert werden.

4.1.2 LINIEN

Viele Messpunkte direkt hintereinander werden oftmals als Linie zusammengefügt. So können GPS-Tracks zu einer Trajektorie zusammengefasst werden (vgl. Abbildung 11, S. 17). Streng genommen handelt es sich um eine Interpolation einer Vielzahl von einzelnen, möglicherweise gefilterten Messpunkten. Neben der Position können weitere Messungen wie Temperatur, Anzahl an Passagieren (zum Beispiel in

öffentlichen Verkehrsmitteln) oder Beschleunigung beziehungsweise G-Kräfte gemessen und visualisiert werden.

4.1.3 POLYGONE

Geotemporale Polygone sind oft Verwaltungsstrukturen mit einer Quote, zum Beispiel Einwohner/km² zu jeder Zeiteinheit. Zu beachten sind hier die kartographischen Grundregeln, so dürfen keine Absolutzahlen als geographische Flächen abgebildet werden. Veränderung kann hier durch Farbe dargestellt werden.

Eine Ausnahme stellen hier anamorphe Karten dar. Im Unterschied zu den oben beschriebenen Choroplethenkarten dürfen hier keine Raten, sondern es müssen Absolutwerte dargestellt werden (BARFORD & DORLING 2008, S. 68). Mit ihnen verändern sich die Geometrien und die Flächen der Objekte können als Abbild des dargestellten Attributes verglichen werden. Der Vergleich von Flächeninhalten verschiedener irregulärer Geometrien ist jedoch äußerst schwierig, wodurch diese Kartenart meist nur zur Erzeugung eines Showeffektes oder zur Bewusstseinsbildung eines Themas eingesetzt wird.

Abbildung 21 zeigt eine anamorphe Weltkarte, welche die Cholerafälle des Jahres 2004 durch die Fläche der einzelnen Gebiete wiedergibt. Für eine erfolgreiche Interpretation einer anamorphen Karte ist eine gute Kenntnis der Ausgangskarte notwendig, in diesem Fall die Länder der Erde. Verhältnisse und negative Werte können nicht dargestellt werden, weiters werden für jedes abgebildete Objekt Werte benötigt. Die Farben dienen ausschließlich zur Identifikation, sie dürfen nicht als Datenträger interpretiert werden. Bei multitemporalen Daten, zum Beispiel einer Entwicklung über mehrere Jahre, verändert sich mit dem Flächeninhalt bei jedem Zeitschritt auch die Position der einzelnen Objekte geringfügig. Die negativen Auswirkungen dieses Effektes können mit einer Animation minimiert werden.

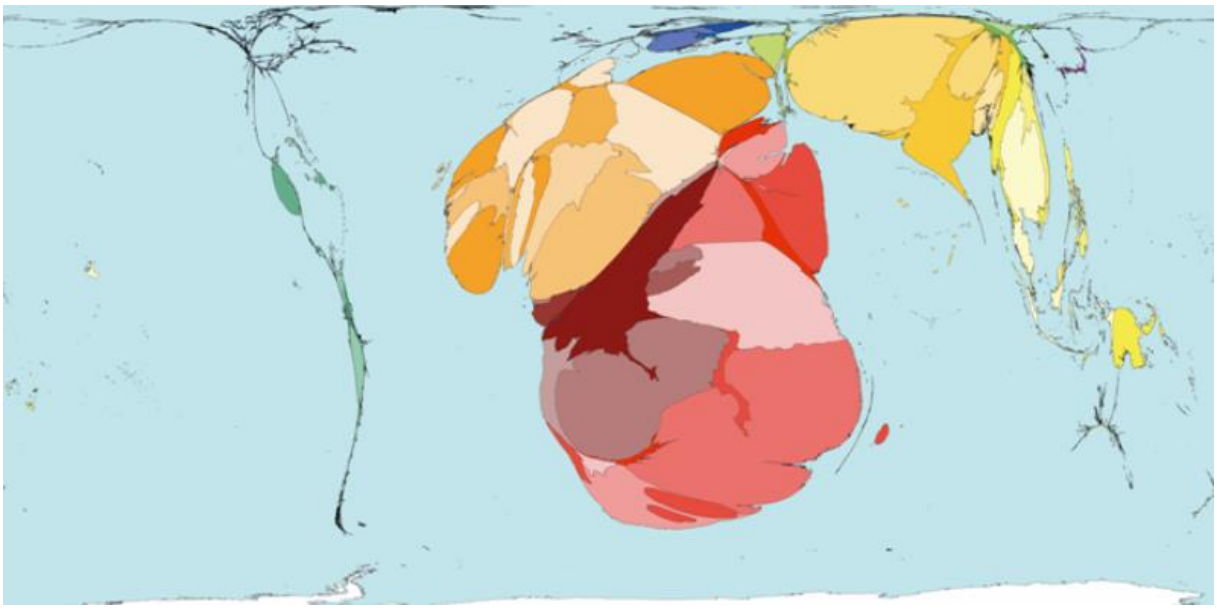


Abbildung 21 Darstellung der Cholerafälle 2004 mittels einer anamorphen Karte. (Quelle: BARFORD et al, slideplayer.com)

4.1.4 GEOMETRIEN MIT 3. DIMENSION

Alle genannten Geometrien treten in der Natur in einer dreidimensionalen Form auf. Meist wird diese durch eine Projektion auf eine Ebene reduziert, vor allem da die Visualisierung von Dreidimensionalität in der Praxis grafische Probleme hervorrufen kann. Eine dreidimensionale Darstellung sollte demnach stets begründet sein und einen direkten Vorteil im Vergleich zu einer simplizierten Visualisierung aufweisen können.

4.2 Datenformate

Allgemein können die geläufigen Datenformate in zwei Kategorien eingeteilt werden: Normen, welche von einer staatlich oder international legitimierten Organisation veröffentlicht werden, und Standards, die von Branchenverbänden erarbeitet werden. (ZIMMERMANN 2012, S. 25)

Zu den wichtigsten Organisationen zählen nach ZIMMERMANN:

- OGC (Open Geospatial Consortium)
  Das OGC ist ein internationales Industriekonsortium bestehend aus über 526 Firmen, staatlichen Behörden und Universitäten. Ihre Aufgabe ist es, gemeinsam interoperable Lösungen für ein „geo-enabled“ Web, location-based Services sowie mainstream IT zu erarbeiten. Die Standards der OGC sollen es Technologieentwicklern ermöglichen, komplexe räumliche Informationen sowie Services für weitere Applikationen zugänglich und verwendbar zu machen.[43] Die GI-Standards werden als OGC-Dokumente kostenfrei im Internet

[43] OGC (2017): OGC. http://www.opengeospatial.org/ogc

angeboten und sind somit für jedermann frei verfügbar (ZIMMERMANN 2012, S. 25).

- ISO (International Organisation for Standardization)
  ISO ist ein Zusammenschluss nationaler Normierungsorganisationen, unter anderem auch dem Deutschen Institut für Normierung e.V. (DIN) sowie dem Austrian Standards Institute (ASI, früher: Österreichisches Normungsinstitut ON[44]). Zuständig für die Normierung im Bereich Geoinformatik ist ISO/TC211[45]. Ihre Aufgabe ist die Standardisierung im Bereich digitaler geographischer Information, dies beinhaltet Methoden, Werkzeuge, Services für Datenmanagement, Erwerb, Verarbeitung, Analyse, Zugriff, Präsentation sowie Austausch von digitalen Geodaten .

Neben den geregelten Standards und Normen sind in der Geoinformatik noch weitere, nicht standardisierte Datenformate geläufig. Diese werden als de-facto Standards oder Industriestandards bezeichnet. Ein Beispiel hierfür ist das weit verbreitete Shapefile-Format (shp) von ESRI (Environmental Systems Research Institute). Es ist jedoch durchaus üblich, dass bewährte Normen von staatlichen oder internationalen Organisationen als echte Standards übernommen werden. Ein Beispiel hierfür wäre KML[46] von Google.

Weiters können Geodaten in Raster- und Vektordaten gegliedert werden. Im Zuge dieser Masterarbeit wird lediglich auf Vektordaten als primärer Informationsträger Rücksicht genommen, Rasterdaten werden nur als „Basemap“ zur Orientierung behandelt.

4.2.1 SHP

Das von ESRI entwickelte Shapefile-Format ist eines der geläufigsten Datenformate im GIS-Bereich. Es besteht aus mehreren Dateien, mindestens jedoch aus .shp (Geometrie), .dbf (dBASE-Tabelle für zusätzliche Attribute) und .shx (Indices der Feature-Geometrien) [47] . Es ist ein sehr mächtiges Format und kann spezielle geographische Informationen wie Projektionseigenschaften, Geocoding oder Metadaten mitliefern. Shapefile gilt als de-facto Standard.

4.2.2 GEOJSON

GeoJSON basiert auf dem weit geläufigen JSON (JavaScript Object Notation) Format und kann somit auch mit einer Vielzahl an Variablen erweitert werden. Diese Variablen

[44] ISO (2017): Members. https://www.iso.org/members.html
[45] ISO (2017): ISO/TC211. https://www.iso.org/committee/54904.html
[46] OGC (2017): KML http://www.opengeospatial.org/standards/kml
[47] ESRI: Shapefile file extensions
http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?topicname=Shapefile_file_extensions

werden im JSON-Attribut „properties“ gespeichert und können vom Datentyp „number“, „string“, „boolean“, „null“, „array“ und „object“ sein.[48]

Code 4 einfaches GeoJSON eines Punktfeatures mit Attributen verschiedener Datentypen (string, number, boolean und array)

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [15.4, 47.0]
  },
  "properties": {
    "name": "Graz",
    "einwohner": 320500,
    "statutarstadt": true,
    "autobahnen": ["A2", "A9"]
  }
}
```

Ursprünglich unterstützte GeoJSON auch ein „crs”-Attribut (Coordinate Reference System), doch dieses wurde von der GeoJSON-Spezifikation im Jahr 2014 gestrichen. Zwar ist die Angabe eines crs-Codes immer noch als herkömmliches Attribut möglich, jedoch sollte nicht erwartet werden, dass jeder Webdienst einen Zugang zu EPSG-Datenbanken (European Petroleum Survey Group) zur Konvertierung besitzt. Es wird daher dringend empfohlen, das Standard-CRS 4236 (WGS 84) zu verwenden.[49] Wie das Shapefile Format gilt GeoJSON als de-facto Standard.

4.2.3 GML

Die Geographic Markup Language (GML) ist eine XML (Extensible Markup Language)-Grammatik zum Austausch von räumlichen Daten und ihren Attributen. Neben Vektordaten können auch Raster- und Sensordaten, sowie CRS, Topologien, Zeit und Regeln zur Kartendarstellung beschrieben werden. GML ist sowohl OGC als auch ISO Standard.[50]

4.2.4 KML

Googles KML (Keyhole Markup Language) ist eine XML-Grammatik und ein von OGC akzeptierter Standard zur Beschreibung von Geodaten. Der Fokus von KML liegt auf der Visualisierung von Geodaten, insbesondere das Inkludieren von Beschriftungen und Bildern, sowie die Navigation von Benutzeransichten wie Kameraflüge und

[48] MACWRIGHT, T. (2015): More than you ever wanted to know about GeoJSON. https://macwright.org/2015/03/23/geojson-second-bite.html#projections
[49] GILLIES, S. (2014): Pruning CRS from GeoJSON. https://sgillies.net/2014/08/06/pruning-crs-from-geojson.html
[50] OSGeo (2017): Geography Markup Language (GML) https://live.osgeo.org/de/standards/gml_overview.html

Blickwinkel. KML wirkt komplementär zu anderen OGC-Standards wie GML, WFS (Web Feature Service) und WMS (Web Map Service).[51]

4.2.5 GPKG

GeoPackage (GPKG) ist ein offener von OGC akzeptierter Geodaten-Standard. GeoPackage ist eine standardisierte Art und Weise, wie Geodaten in eine SQLite-Datenbank gespeichert werden. In dieser können sowohl Vektor- wie auch Rasterdaten verspeichert werden sowie Attribute und zusätzliche Erweiterungen.[52]

GeoPackage wird häufig als möglicher Nachfolger des Shapefile-Formats gehandelt. Es erledigt ähnliche Aufgaben, verpackt jedoch alle Daten in nur einem einzigen, simplen File. Da es ein hersteller- und plattformunabhängiges Datenformat ist und besonders auch für den inzwischen immer wichtiger werdenden mobilen Sektor konzipiert wurde, könnte dieses Format in Zukunft eine größere Bedeutung haben. Momentan wird der Markt bei Desktopanwendungen vom Shapefile dominiert.

4.2.6 CSV

CSV (Comma Seperated Values) ist ein weit verbreitetes Datenformat zum Austausch von Daten in Tabellenform (spreadsheets). Obwohl es bereits relativ alt und einfach ist und sehr viel Anwendung findet, wurde es nie formell dokumentiert. Daher existieren viele unterschiedliche Implementationen.[53]

Dies betrifft

- Separator (`,` oder `;`)
- Zeilenende (`\r\n` auf Windows-Systemen, `\n` auf UNIX-Systemen)
- Header

CSV hat keinerlei direkten Bezug zu Geodaten. Da CSV-Dateien jedoch auch für Laien auf jedem Betriebssystem und mit minimaler Software erstellbar sind, können simple geographische Datensätze, wie etwa Koordinatenpaare und Attribute, manuell oder mit einem Tabellenkalkulationsprogramm erstellt werden. Zusätzlich können CSV-Dateien relativ einfach automatisiert oder halbautomatisiert in einer Vielzahl von Programmierumgebungen gebildet werden.

4.2.7 WELCHE DATENFORMATE UNTERSTÜTZEN?

Bevor man sich für die Unterstützung eines Datenformates für sein eigenes Projekt entscheidet, sollte man sich einiger grundlegenden Probleme bewusst sein. Diese

[51] WILSON, T. (2007): KML 2.2 – An OGC Best Practice. https://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/%3fartifact_id=23689
[52] OGC (2017): GeoPackage. http://www.geopackage.org/
[53] SHAFRANOVICH, Y. (2005): Common Format and MIME Type for Comma-Separated Values (CSV) Files. https://tools.ietf.org/html/rfc4180#page-2

betreffen hauptsächlich die Ansprüche der Benutzerseite sowie die technische Umsetzbarkeit.

Wird eine frei zugängliche Applikation für das Web entwickelt, so sollte klar kommuniziert werden, wofür das Programm gebraucht werden soll. Der Benutzer muss genau wissen, was das Programm kann, wie es benutzt werden muss, welche Daten geliefert werden müssen und welche Fehler entstehen können.

Werden die Daten in die Applikation falsch eingespeist oder werden falsche Daten in das Programm geladen, so können zweierlei Fehler auftreten:

- Technische Fehler
  Bei technischen Fehlern können die vom Benutzer gelieferten Daten vom Programm nicht gelesen beziehungsweise nicht verwertet werden. Eine aussagekräftige Fehlermeldung als Feedback ist vom Programm zurückzugeben.
  Technische Fehler können folgendermaßen auftreten:
  - der Benutzer hat das verlangte Datenformat nicht richtig umgesetzt
  - das Datenformat ist in der Applikation nicht korrekt implementiert
  - es gab Probleme bei der Datenübertragung
  - der Benutzer hat versucht, eine falsche Datei einzulesen
- Inhaltliche Fehler
  Auch für Karten im Web gilt: „Eine schlecht gemachte Karte kann ihre Geschichte manchmal erfolgreich erzählen, aber eine elegant erstellte Karte gemacht aus schlechten Daten kann schlechter als nutzlos sein, sie kann irreführen“ (TYNER 2015, S. 5).
  Inhaltliche Fehler sind gefährlicher als technische Fehler, da ein Abbruch des Programms nicht automatisch geschehen kann und der Benutzer daher nicht vor möglichen Missinterpretationen gewarnt werden kann.
  Möglichkeiten, inhaltliche Fehler vorzubeugen, sind aussagekräftige Tutorials, wie die Software benutzt werden muss, eine klare Kommunikation, welche Art von Daten unterstützt werden sowie, soweit möglich, das interne Abfangen von Fehlern durch Programm selbst, welches wiederum Feedback an den Benutzer zurückgeben muss.
  So ist es bei einem Programm, welches ausschließlich mit intervallskalierten Zahlen funktionieren soll, elementar, einen Fehler auszugeben, sollte der Benutzer versuchen, Buchstaben einzugeben. Jedoch sind nicht alle Daten, welche durch Zahlen wiedergegeben werden können, intervallskaliert (Beispiel: Schulnoten). Dadurch kann das vom Programm verlangte Datenformat zwar eingelesen und verarbeitet werden, jedoch ergeben viele Rechenoperationen

inhaltlich keinen Sinn. Eine Fehlinterpretation der Ergebnisse ist in diesem Fall wahrscheinlich.

Auch die Wahl der unterstützten Datenformate kann inhaltliche Fehler vorbeugen. Unterstützt man echte Geodatenformate wie Shapefile, GeoJSON oder GML, so ist die Wahrscheinlichkeit fehlerhafter Koordinaten (wie etwa falscher Kommaseperator oder falsche Reihenfolge der geographischen Länge und Breite) geringer, da diese Datensätze meist mit GIS-Software erstellt wurden.

4.2.7.1 Welche Datentypen müssen unterstützt werden?

Als Informationsträger von Daten werden in dieser Masterarbeit nur Vektorformate berücksichtigt. Die clientseitige Bearbeitung und Visualisierung von Rasterdaten ist wesentlich aufwändiger und fehleranfälliger. Außerdem sind selbst einfache Rechenoperationen mit Rasterdaten, wie etwa das oftmals notwendige Umprojizieren, um ein vielfaches rechenintensiver als das Umprojizieren von Vektordaten. Dies wirkt sich negativ auf die geforderte Hardware des Clients und die User Experience aus.

Geocodierte Rasterdaten werden vor allem von Fachexperten aus der Fernerkundung verwendet. Ihnen kann eine inhaltlich sinnvolle Vorprozessierung und Transformation in ein Vektorformat zugemutet werden. Ausnahmen stellen hier Fernerkundungsdaten aus Drohnenflügen dar, welche auch im Amateurbereich immer mehr Anwendungen finden werden.

4.2.7.2 Wer ist meine Zielgruppe?

Hat der Benutzer gar keine Vorkenntnisse in der Geoinformatik, so müssen die Daten in einer nicht-GIS-Software aufbereitbar sein. Fast alle Daten haben einen Raumbezug, es wäre nicht ratsam, ein simples Programm zur Datenvisualisierung nur Fachexperten verfügbar zu machen. Vermutlich ist das Gegenteil der Fall: Experten aus der Geoinformatik haben meist Zugriff auf elaborierte Visualisierungs- und Datenbankensoftware, die Notwendigkeit einer Browserapplikation mit eingeschränkten Funktionalitäten sinkt mit höherer vorhandener Infrastruktur.

4.2.7.1 In welchem Datenformat liegen die Ausgangsdaten vor?

Die Frage nach den Ausgangsdaten des Benutzers ist wohl die wichtigste. Handelt es sich beim Benutzer um den Auftraggeber einer Spezialapplikation, welche nur für seine Bedürfnisse zugeschnitten werden soll, erübrigt sich die Frage des Datenformats, man wählt entweder das bereits vorhandene Format, oder - falls dieses nicht direkt verarbeitbar ist - ein Format, welches für den Auftraggeber einfach zu kreieren und verarbeitbar ist. Da das Ziel dieser Masterarbeit jedoch eine Webapplikation für ein heterogenes Zielpublikum ist, sollte das Datenformat einen kleinsten gemeinsamen Nenner aller möglichen Benutzeranforderungen darstellen.

Heutzutage können viele Geodaten im Browser über Drag-and-Drop-Applikationen ohne Datenupload konvertiert werden. So kann unter anderem das weit verbreitete Shapefile Format in einem Desktop-GIS von Experten erstellt und im Browser in GeoJSON, GeoPackage oder CSV umgewandelt werden.[54]

4.2.7.2 Das Datenformat als offener Standard?

Ist das Datenformat ein offener Standard, so kann Interoperabilität mit anderer Software besser gewährleistet werden. Außerdem unterstützen in der Geoinformatik die meisten Softwarepakete eine Vielzahl von offenen Standards, daher ist es oft möglich, Daten in einem passenden offenen Standard zu exportieren.

4.2.7.3 Wie soll auf die Daten zugegriffen werden?

Es bestehen verschiedene Möglichkeiten, wie eine Webapplikation auf die Daten des Benutzers zugreifen kann (mehr dazu im Kapitel 4.4: Darstellung geotemporaler Daten: Status Quo).

Soll die Webapplikation rein browserseitig agieren, müssen die Geodaten vom Browser verarbeitbar sein. Dazu zählen die meisten textbasierten Formate (GeoJSON, GML, KML, CSV), während Binärformate (SHP, GPKG) nicht direkt verarbeitet werden können.

Im Rahmen dieser Masterarbeit wurde in Anbetracht aller oben genannten Faktoren entschieden, die Datenformate GeoJSON und CSV zu unterstützen.

GeoJSON kann als offener, textbasierter Standard aus den meisten gängigen GIS-Paketen exportiert und im Browser durch das native JSON-Format besonders schnell verarbeitet werden.

CSV kann auf allen Systemen aus Spreadsheet-Programmen wie Microsoft Excel oder OpenOffice Calc exportiert und im Bedarf mit einem beliebigen Texteditor nachbearbeitet werden. CSV ist daher besonders für Laien interessant. Die Erstellung eines Datensatzes mit Koordinaten ist zwar etwas aufwändiger, oftmals ändern sich die Koordinaten jedoch nicht und es kann der Datensatz mit neuen Daten erweitert oder aktualisiert werden.

Da für CSV keine Standardisierung für Koordinaten existiert, muss das verlangte Datenformat dem Benutzer klar kommuniziert werden. Beispieldaten und Tutorials sind dafür die besten Optionen.

[54] MYGEODATA: https://mygeodata.cloud/converter/shp-to-geojson

4.3 Darstellung multivariater Daten

4.3.1 CHERNOFF-GESICHTER

Die sogenannten „Chernoff-Gesichter“ wurden erstmals 1973 von Herman Chernoff vorgestellt. Es handelt sich dabei um eine Möglichkeit, multivariate Daten und komplexe Zusammenhänge einfach und plakativ darzustellen. Als Symbole dienen hierbei abstrahierte Gesichter, die Gesichtszüge repräsentieren die Ausprägungen der einzelnen Variablen. So können mit Form, Größe und Position der Augen, Nase, Mund, Haare, etc. bis zu 15 verschieden Variablen mit jedem Gesicht abgebildet werden. (TYNER 2010, S. 182)

Mit dieser Art der Visualisierung multivariater Daten können jedoch auch Probleme auftreten. Durch den Gebrauch von Gesichtern werden die Symbole wertend. Jeder Mensch assoziiert mit einem Gesicht gewisse Emotionen. Manche können bewusst gesteuert werden, ein lachendes Gesicht wird meist positiv aufgefasst, ein trauriges negativ. Weisen die Augenbrauen nach innen, wirkt das Gesicht böse, sind sie nach außen geneigt, wirkt es sanft. Andere ausgelöste Assoziationen können vom Kartographen nicht vorausgesehen werden, vielleicht ähnelt ein Gesicht einer Person, mit der man schlechte Erinnerungen verknüpft.

Auch der Einsatz von Farbe ist im Zusammenhang mit Gesichtern oft nicht ratsam. Farbe ist aber eine der besten graphischen Variablen, da sie sofort ins Auge springt und sehr gut verglichen werden kann. Benutzt man natürliche Farben, also Brauntöne verschiedener Helligkeiten, liegt eine Assoziation mit ethnischen Gruppierungen nahe, im Extremfall können Vorurteile des Betrachters eine korrekte Interpretation der Daten beeinflussen. Wählt man bewusst eine unnatürliche Farbgebung, wie eine Farbskala von Grün nach Violett, so wirken die Gesichter wie Außerirdische und die Karte verliert an Ästhetik. (nach TYNER 2010, S. 182)

Ein weiteres Problem ist die Komplexität der multivariaten Symbole. Die theoretisch mögliche Anzahl von 15 Variablen pro Symbol ist schwierig zu ergreifen, selbst wenn man sich nur auf zwei oder drei Faktoren fokussiert. Noch schwieriger wird der Einsatz in einer Karte, welche bereits an sich komplexe Grafiken darstellt.

Abbildung 22 zeigt eine der bekanntesten Anwendungen der Chernoff-Gesichter in der Kartographie: die Karte des Lebens in Los Angeles von Eugene Turner aus dem Jahre 1977, welche sozioökonomische Faktoren in den einzelnen Bezirken der Stadt darstellt. Mit nur vier abgebildeten Variablen mit jeweils nur drei Klassen stellen die Gesichter eine viel geringere Breite an Daten als möglich dar, dadurch bleibt die Grafik les- und interpretierbar.

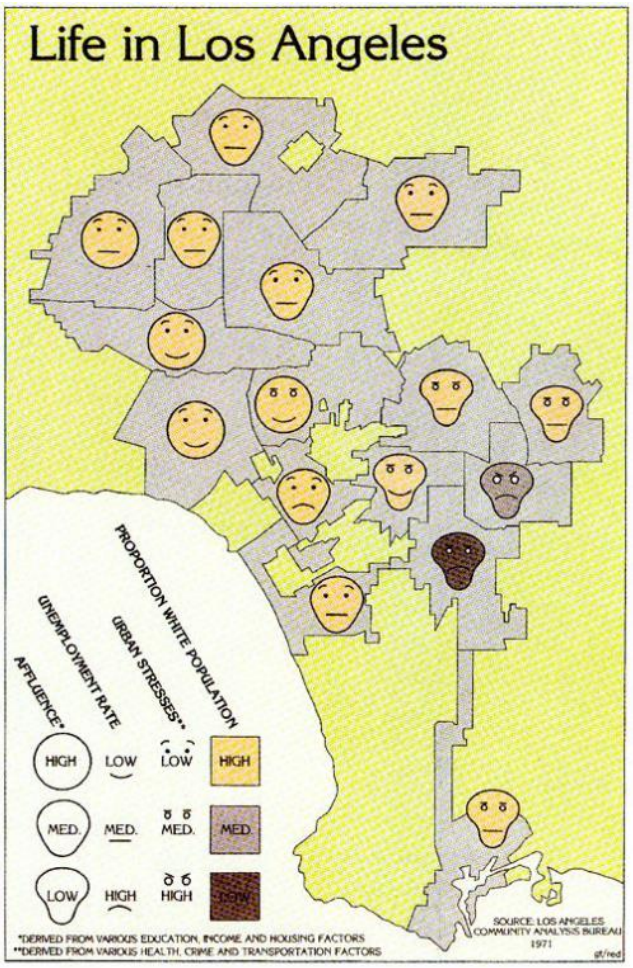


Abbildung 22 Verteilung sozialer Faktoren im urbanen Los Angeles mittels Chernoff-Gesichtern (Quelle: TURNER, E. 1977, in: FIELD, K. (2014): Life in Los Angeles by Eugene Turner icaci.org)

Zu beachten ist hier die Wahl der einzelnen Faktoren und durch welche Gesichtsmerkmale sie dargestellt werden:

- Form: Wohlstand (rund = reicher, schmal = ärmer)
- Mund = Arbeitslosigkeit (lachend = geringe Arbeitslosigkeit, traurig = hohe Arbeitslosigkeit)
- Augen: Urbaner Stress (sanfter Blick = gering, böser Blick = hoch)
- Hautfarbe: Anteil weißer Population (heller = höherer Anteil, dunkler = geringerer Anteil)

Die korrekte Zuweisung der Variablen zu den Gesichtszügen ist essentiell für die Qualität der Karte [55]. Die Variablen wurden von TURNER jenen Gesichtszügen zugewiesen, mit denen sie mit hoher Wahrscheinlichkeit ohnehin assoziiert werden, dadurch werden die Symbole intentional korrekt gedeutet. Eine Karte solcher sozioökonomischen Faktoren ist eine von wenigen Anwendungen, bei der eine solche

[55] FIELD, K. (2014): Life in Los Angeles by Eugene Turner. http://mapdesign.icaci.org/2014/12/mapcarte-353365-life-in-los-angeles-by-eugene-turner-1977/

natürlich richtige Interpretation möglich ist. Meist sind die Variablen abstrakter und können nicht direkt mit Gesichtszügen in Verbindung gebracht werden, wodurch die Symbolik erst gelernt werden muss.

Auch die beim Betrachter ausgelösten Emotionen stellen in diesem Fall kein Problem dar, da geringer Wohlstand und hohe Arbeitslosigkeit generell als negativ angesehen werden können. Eine Möglichkeit, diese Emotionen bei abstrakteren Daten zu reduzieren, ist das Prinzip der Chernoff-Gesichter auf eine andere Grundform umzulegen. Der Mensch ist ein Experte im Erkennen von Gesichtern, jedoch kann diese Fähigkeit auch auf andere bekannte Objekte angewandt werden. Einen interessanten Ansatz verfolgen hier RODRIGUEZ und KACZMAREK (2016, S. 406), indem sie die Vorteile der multivariaten Darstellung der Chernoff-Gesichter auf eine wertungsfreiere Grundform, in diesem Fall einen Fisch, anwenden.

Abbildung 23 zeigt eine Legende zur Anwendung des Chernoff-Fisches zur Darstellung von Attributen von Hedge-Fonds. Die Datengrundlage ist viel abstrakter als sozioökonomische Strukturen und sie erlaubt keine natürliche Assoziation mit menschlichen Gesichtszügen. Ohne den Einsatz von Farbe ermöglicht diese Symbolik die Darstellung von zehn Variablen.

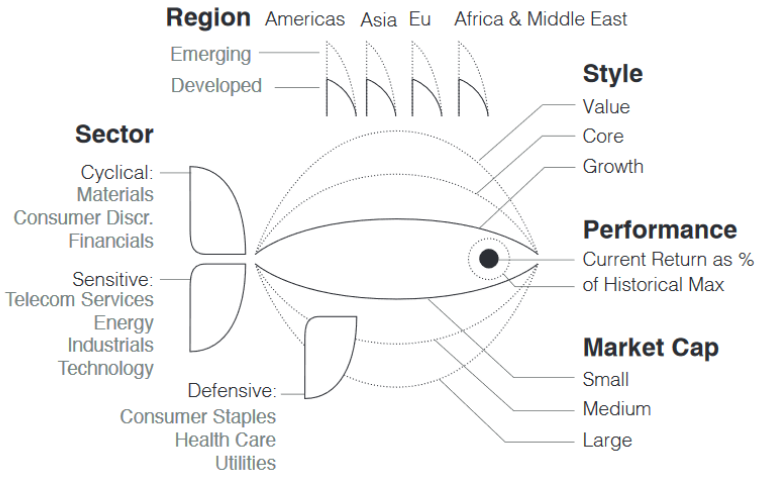


Abbildung 23 Chernoff-Fisch zur Darstellung von Attributen von Hedge-Fonds. (Quelle: RODRIGUEZ & KACZMAREK 2016, S. 406)

Wie die Gesichter ist auch der Fisch kein neutrales Symbol, sondern stellt ein reales Objekt dar. Es muss dem Betrachter klar kommuniziert werden, wie die Grafik zu lesen ist und, dass die dargestellte Thematik nichts mit echten Fischen zu tun hat. Das Prinzip der Chernoff-Gesichter kann auch auf andere abstrahierte Symbole angewandt werden, am besten mit Bezug zur Thematik. Mögliche Grundformen wären Autos, Flugzeuge, Häuser oder Körperformen.

Technisch gesehen ist die Erstellung solcher multivariaten Symbole im Web mit den in Kapitel 2.2: *Webbasierte Visualisierung von Daten ohne Raumbezug* vorgestellten

Technologien Canvas und SVG möglich. Es existieren unter anderem frei verfügbare Plugins für die SVG-Bibliothek D3.js, sowohl für Chernoff-Gesichter[56] als auch für die Variante der Chernoff-Fische[57].

4.4 Darstellung geotemporaler Daten: Status Quo

4.4.1 KARTENREIHEN

In manchen Situationen müssen geotemporale Daten ohne Interaktionen, wie einen Timeslider, dargestellt werden. Eine mögliche Lösung ist der Einsatz von Kartenreihen oder „Small Multiples“. Darunter versteht man eine Zusammenstellung aus mehreren, ähnlichen thematischen Karten, wobei jede Karte einen Schnappschuss zu einem Zeitpunkt darstellt. Für eine klare narrative Struktur werden die einzelnen Karten in chronologischer Reihenfolge abgebildet.[58]

Abbildung 24 zeigt einen Ausschnitt einer in Microsoft Excel erstellten Kartenreihe zur Visualisierung neuer Walmart-Stores in den USA. Dieselbe Grafik ist auch als animiertes GIF vorhanden, jedoch ohne Interaktionsmöglichkeit mit der Variablen „Zeit“ (wie etwa pausieren und untersuchen) ist es nicht möglich, die einzelnen Karten zu vergleichen[59]. Durch Animation können Trends in geotemporalen Daten leichter erkannt werden, durch Kartenreihen beziehungsweise durch Pausieren einer Animation können genaue Gegenüberstellungen besser vollzogen werden.

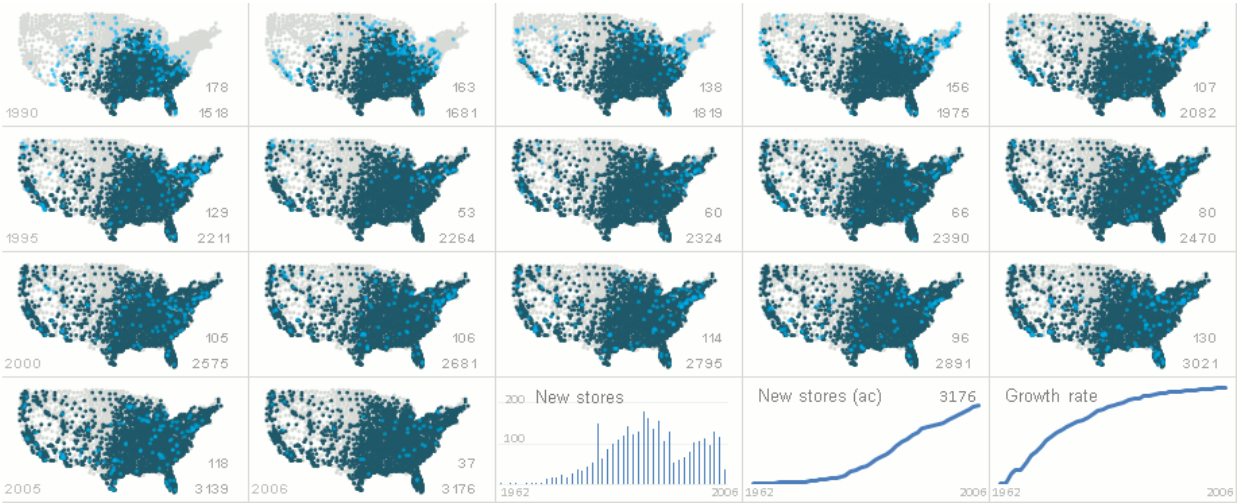


Abbildung 24 Ausschnitt einer Visualisierung jährlich neuer Walmart-Stores in den USA. Hellblaue Punkte repräsentieren neue Stores, dunkelblaue Punkte die bereits existenten. (Quelle: excelcharts.com)

[56] SINGH, G. (2015): Chernoff Faces with D3 and React.js https://kilotau.com/chernoff-faces-with-d3-and-react-js/
[57] MEAGHER, T. (2016): Chernoff Fish https://github.com/tmm/chernoff-fish
[58] FHP (2008): Small Multiples. http://geo.patternbrowser.org/code/pattern/pattern_anzeigen.php?4,298,17,0,0,319
[59] CAMOES, J (2012): Animation, Small Multiples or the Reorderable Matrix? Growth of Walmart, Excel Edition. https://excelcharts.com/animation-small-multiples-growth-walmart-excel-edition/

4.4.2 ANIMIERTE KARTEN

Animierte Karten erzeugen eine Illusion der kontinuierlichen Veränderung. Diese Illusion wird durch das schnelle Darstellen einzelner Karten erzeugt, so schnell, dass das Auge die einzelnen Bilder nicht mehr als solche wahrnimmt. Dieser Effekt tritt bei etwa 24-30 Frames, also Einzelbilder, pro Sekunde auf (HARROWER & FABRIKANT 2008 S. 50).

Bereits im zweiten Weltkrieg wurden mit animierten Karten Truppenverschiebungen dargestellt, jedoch war die Produktion solcher Animationen äußerst zeitaufwändig und benötigte teures Spezialequipment (TYNER 2015, S. 100). Heute sind Animationen für das Web relativ einfach zu erstellen, sogar ohne Plugins wie Adobe Flash können heute Animationen über Webstandards wie CSS oder Canvas nativ in jedem Browser umgesetzt werden.

Grundsätzlich können animierte Karten zwei Sachverhalte darstellen: entweder eine Veränderung der Daten abhängig von der Zeit oder eine Veränderung der Daten abhängig von einem Attribut. Zu Letzterem zählen meist auch digitale Überflüge sowie Zoom- und Pan-Animationen, da davon ausgegangen wird, dass die Karte sich nicht zeitlich, sondern nur die virtuelle Position des Betrachters verändert. Im Zuge dieser Masterarbeit werden jedoch nur Animationen zeitlicher Veränderungen berücksichtigt.

Der Nachteil von animierten Diagrammen oder Karten ist die Schwierigkeit, die einzelnen Zeitschritte direkt zu vergleichen. Karten sind besonders komplexe Grafiken, meistens ist es nicht möglich, sich die Momentaufnahmen jedes Zeitschrittes zu merken. Während die Masse an Daten, die mit Animationen digital dargestellt werden kann, nahezu unbegrenzt ist, hat der Benutzer hingegen nur sehr begrenzte Fähigkeiten, Informationen aus den Daten zu generieren und im Kurzzeitgedächtnis zu speichern (HARROWER&FABRIKANT 2008, S. 54). Eine Lösung hierfür wären mehrere Einzelkarten („Small Multiples“) hintereinander, um eine Vergleichbarkeit zu gewährleisten.[60]

Aufgrund der dadurch stark verringerte Größe von statischen Small Multiples, die zur gleichzeitigen Betrachtung der einzelnen Grafiken erstellt wurden, müssen auch die dargestellten Informationen reduziert werden. Bei interaktiven Karten kann die Erstellung von Snapshots zu gewünschten Zeitpunkten einen akzeptablen Kompromiss darstellen zwischen der intuitiveren Wahrnehmung von zeitlichen Mustern aus einer flüssigen Animation und der einfacheren Vergleichbarkeit und Auffassbarkeit von statischen Karten, ohne den Karteninhalt zu limitieren.

[60] EEA (2016): Chart dos and don’ts https://www.eea.europa.eu/data-and-maps/daviz/learn-more/chart-dos-and-donts

Insbesondere bei großen Datenreihen kann so durch Input des Benutzers ein Mehrwert generiert werden. Entweder weiß der Benutzer, zum Beispiel durch Expertenwissen, bereits im Vorhinein, welche Zeitpunkte er vergleichen möchte, oder er benutzt die Animation beziehungsweise andere Interaktionen, um sich einen groben Überblick über die Daten zu verschaffen und stellt anschließend besonders interessante Zeitpunkte gegenüber.

4.4.3 SPACE-TIME CUBE

Ein Space-Time Cube ist in seiner einfachsten Form ein Würfel, wobei die X- und Y-Achse der Basis eine Repräsentation der Geographie darstellt (zum Beispiel geographische Länge und Breite) und die Z-Achse die Zeit verkörpert. Der Space-Time Path, der durch die Visualisierung von durchgehenden geotemporalen Daten in einem Space-Time Cube entsteht, kann auf die Basis projiziert werden, wobei ein Fußabdruck des Pfades auf der Karte entsteht. Eine weitere Variable kann durch die Dicke beziehungsweise durch die Farbe des Pfades visualisiert werden. (nach KRAAK 2008, S. 395ff).

Abbildung 25 zeigt ein häufiges Thema für den Space-Time Cube: Napoleons Marsch nach Russland im Jahre 1812. Anhand der vertikalen Abschnitte im Space-Time Pfad können die Stationen identifiziert werden, in denen eine zeitliche Veränderung, aber keine Veränderung im Raum geschah, also ein Stillstand der Truppenbewegungen. Die Breite des Pfades repräsentiert Truppenstärke. Die geographische Basis der X-Y-Ebene spielt in dieser Version eine untergeordnete Rolle, eine einfache Karte ohne Labels zeigt die Route als Projektion des Pfades.

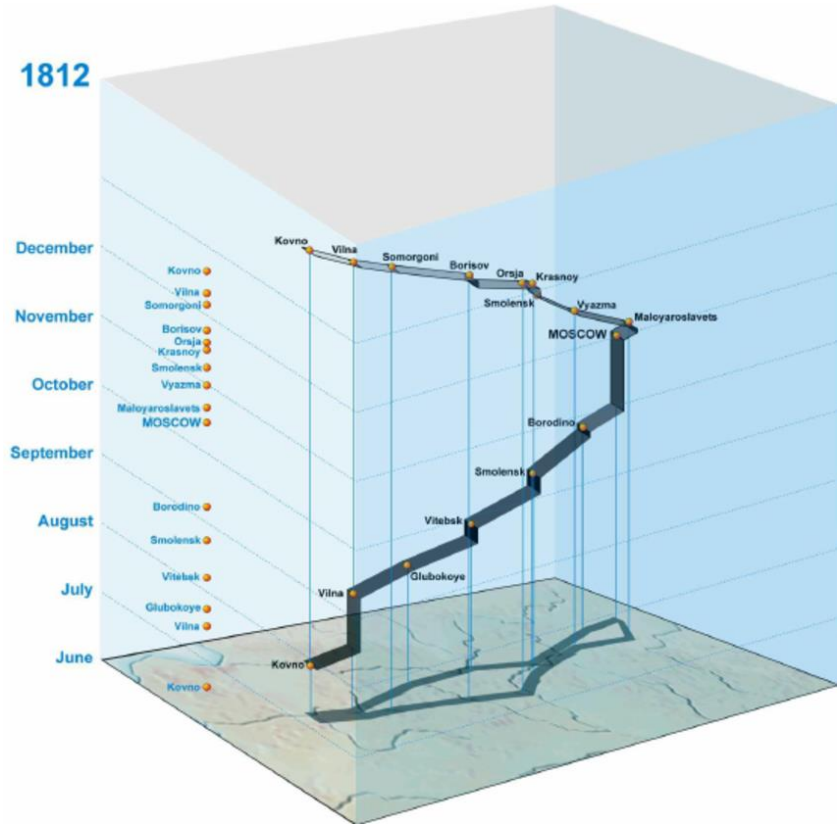


Abbildung 25 Napoleons Marsch nach Russland in einem Space-Time Cube nach KRAAK, die Dicke des Pfades repräsentiert die Truppenstärke (Quelle: NÖLLENBURG 2006)

Diese Grafik hat zwei Hauptaussagen: Zum einen den langen Aufenthalt der Truppen vor Moskau, zum anderen die Abnahme der Truppenstärke über die Zeit. Diese Aussagen werden dem Betrachter ansehnlich übermittelt, jedoch hat diese Darstellungsform einige gravierende Nachteile, besonders bei einer allgemeineren Datenlage als der hier abgebildeten.

Die Darstellung von Bewegung eines einzigen Akteurs wie in Abbildung 25 gelingt mit einer akzeptablen Qualität. Schwieriger wird es bei mehreren Akteuren, da die einzelnen Pfade im dreidimensionalen Raum schwieriger zu differenzieren werden. Selbst bei der Visualisierung von Napoleons Truppenbewegung wurde diesbezüglich nachgeholfen. Eigentlich handelt es sich um mehrere kleinere Truppen, die sich auf dem Weg nach Moskau sammeln und zu einem einzigen Verband zusammenschließen.

Die redundante Benennung der Ortschaften ist suboptimal, eine Labelung auf der Karte der Basisebene wäre klarer. Auch die Zeit, ein zentrales Thema bei der Darstellung geotemporaler Daten, ist in diesem Beispiel, wenn überhaupt, nur durch Umwege erkennbar. Durch die Verzerrung durch den Ansichtswinkel kann die Zeit auf dem Pfad nicht direkt abgelesen werden. Diese Verzerrung verhindert auch eine sichere Deutung der vierten Variablen, der Truppenstärke. Man sieht zwar, dass diese

mit zunehmender Zeit deutlich abnimmt, jedoch kann durch die Dreidimensionalität selbst mit einer Legende keine klare Zahl eindeutig bestimmt werden.

Eine Möglichkeit diese Mängel auszugleichen, ist ein interaktiver, multivariater Space-Time Cube, bei dem die Basisebene verschoben werden kann. Diese weiterentwickelte Variante weist eine klarere Möglichkeit auf, die Zeit und die Anzahl der Soldaten abzubilden. Letztere wird in Abbildung 26 über ein Histogramm auf der t-Achse visualisiert. Selbst die Lufttemperatur ab dem Rückzug findet als zusätzliche Variable an einem Eckpunkt der Datenebene Platz. (nach TRAN & NGUYEN 2012)

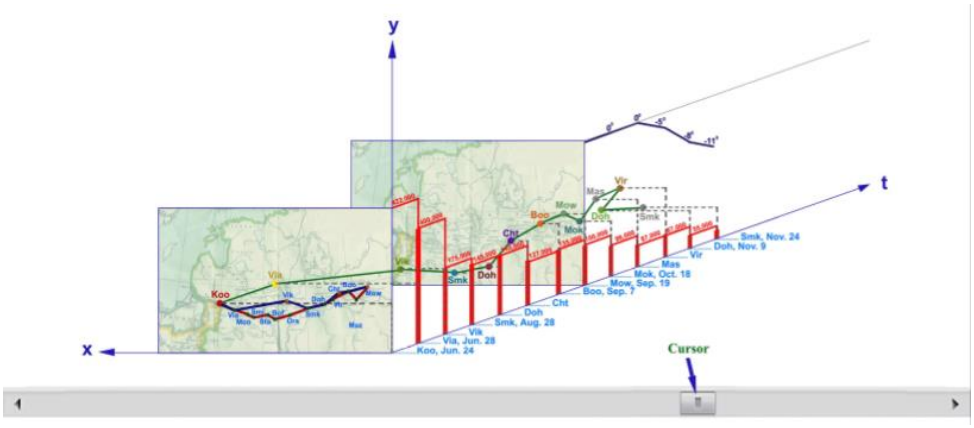


Abbildung 26 Napoleons Truppenbewegungen nach Moskau, visualisiert in einem interaktiven, multivariaten Space-Time Cube (Quelle: TRAN & NGUYEN 2012)

5 CLIENTSEITIGE DATENVERARBEITUNG

Das heutige Spektrum an clientseitiger Hard- und Software ist auf dem ersten Blick äußerst inhomogen. Clientseitige Hardware kann in Desktop und Mobile Devices gegliedert werden. Auf Seiten der Software existieren verschiedene Betriebssysteme (Windows, Macintosh, Linux), wobei auf jedem dieser Betriebssysteme mehrere Browser operieren können. Diese Browser arbeiten großteils nach den Standards des W3C und können daher mit einer einzigen Webapplikation erreicht werden, welche auf allen Systemen gleichermaßen funktioniert. Ist ein Datenformat durch das W3C standardisiert, so sollten alle gängigen Browser mit einheitlichen Funktionen dieselben Ergebnisse liefern.

Ein weiterer Vorteil der clientseitigen Datenverarbeitung ist der Datenschutz. Das Bewusstsein über Datenschutz steigt insbesondere bei personenbezogenen „Big Data“, welche empfindliche persönliche Informationen, Aktivitäten und Verhalten in der physischen und virtuellen Welt widerspiegeln können (YANG et al 2015, S. 331). Auch in der Industrie ist Datensicherheit aufgrund sensibler Daten sowie Industriegeheimnissen ein relevantes Thema.

6 MAPOCH: CLIENTSEITIGE VISUALISIERUNG VON GEOTEMPORALEN MULTIVARIATEN DATEN IN DER PRAXIS

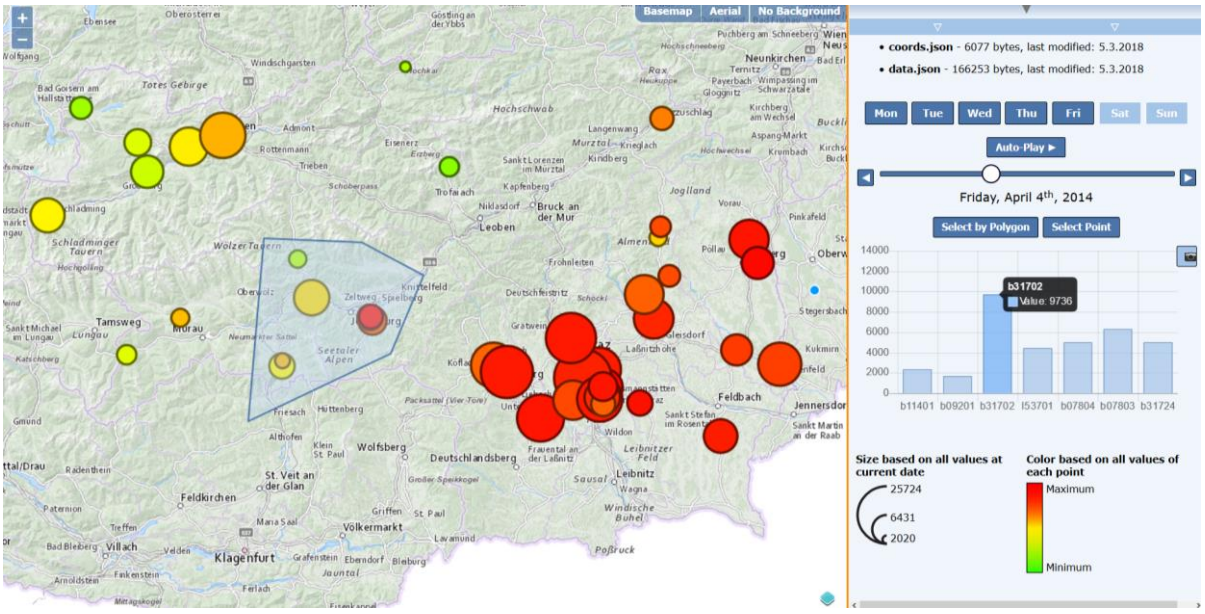


Abbildung 27 Die Grundfunktionen der Applikation Mapoch nach dem Einlesen lokaler Daten (Quelle: eigene Abbildung)

6.1 Die Applikation „Mapoch“: Ziele, Funktionen, Anwendungsgebiete

Mapoch ist ein Kunstwort zusammengesetzt aus den englischen Worten „map“ für Karte und „epoch“, für Epoche, einem in der Geographie geläufigen Ausdruck für einen bestimmten Zeitpunkt in temporalen Daten. Die Grundfunktion der Applikation soll die Darstellung lokaler, geotemporaler Daten in allen gängigen Browsern sein. Die Daten sollen interaktiv erkundbar sein, so dass ein klarer Vorteil gegenüber herkömmlichen Darstellungen der Ausgangsdaten gegeben ist. Zu diesen Darstellungsformen zählen unter anderem eine reine tabellarische Auflistung der Werte und eine Darstellung als Diagramm oder Diagrammreihe ohne Ortsbezug.

Die Zielgruppe leitet sich von der Funktion des Programmes ab. Da die Hauptfunktion die Visualisierung und Präsentation großer Zeitreihen von verschiedenen georeferenzierten Messpunkten ist, richtet sich die Anwendung an alle, die mit einer solchen Datenlage konfrontiert sind. Das sind meist Experten in ihrem Fachgebiet, sowohl mit als auch ohne direkten Geo-Bezug. Somit richtet sich Mapoch auch an Laien der Geoinformatik, einige geographische Grundroutinen müssen dementsprechend vereinfacht kommuniziert werden.

Abbildung 27 zeigt die Grundfunktionen von Mapoch am Beispiel von Verkehrsmessungsdaten inder Steiermark. Es handelt sich um einen Datensatz aus den Messtellen, der mit einem Datensatz der Messwerte verknüpft wird und lokal im Browser eine interaktive Visualisierung generiert. Dadurch, dass für jeden der 44 Messpunkte in der Steiermark ein Messwert für über 250 Tage vorhanden ist, ist der Datensatz sehr unübersichtlich. Des weiteren sind aus der ursprünglichen Tabellendarstellung der Messdaten geographische Zusammenhänge nicht ersichtlich, durch die Visualisierung in Mapoch werden diese auf einem Blick erkennbar.

Um die geotemporale Komponente der Messdaten dem Benutzer näher zu bringen, kann dieser den angezeigten Zeitpunkt über einen Schieberegler bzw. über Pfeilknöpfe verändern. Auch eine Animation mit einem Update zum nächsten Zeitpunkten in den Daten im Sekundentakt ist möglich. Durch weitere Auswahloptionen können einzelne Wochentage ein- beziehungsweise ausgeschalten werden. Daduch kann in diesem Beispiel der Benutzer die Verkehrslage eines bestimmten Tages, zum Beispiel jedes Sonntages, direkter miteinander vergleichen.

Eine weitere Grundfunktion von Mapoch ist die gezeigte Selektion mehrerer Messtellen und Generierung eines Balkendiagramms. Die Darstellung von Mapoch ist zwar nicht in erstere Linier dazu geschaffen, genaue Einzelwerte darzustellen, sondern um einen groben Überblick über die Datenlage zu erhalten, mit dieser Funktion können dennoch die importierten Messwerte exakt angezeigt und miteinander verglichen werden. Die absoluten Messwerte der Verkehrsdaten sind als vertikale Balkendiagramme viel intuitiver erfassbar als durch die Flächeninhalte der Kreise in der Karte.

Um zwei verschiedene Zeitpunkte oder auch verschiedene Selektionen besser vergleichar zu machen, wurde noch die Funktion der Snapshots implementiert. Mit Snapshots werden der aktuelle Zeitpunkt in den Daten und selektierte Elemente gespeichert, dadurch können zum Beispiel drei selektierte Verkehrsmessstellen in einem Dorf Anfang Jänner und Anfang Juni direkt gegenübergestellt werden.

Die Grundfunktionen von Mapoch benötigen demnach statische Messpunkte, sowie einen temporalen Datensatz, der mit den Messpunkten verknüpft werden kann. Der Nutzen für den Benutzer liegt zum einen in einem groben ersten Überblick über seine Datengrundlage, ob die Daten stimmen können, ob gravierende Außreißer oder Messfehler enthalten sind, zum anderen in der Möglichkeit, geotemporale Daten tiefer zu erkunden und Muster oder Regeln sowohl in der geographischen, als auch in der zeitlichen Komponente der Daten zu finden.

Mögliche Anwendungsbereiche sind:

- Verkehr (automatisierte oder manuelle Verkehrszählungen)
- Klima- und Meteorologie (Messungen mehrerer Klimastationen, Temperatur, Niederschlag, Luftfeuchtigkeit, Luftdruck)
- Land- und Forstwirtschaft (zeitabhängige Bodenwerte wie Bodenfeuchtigkeit, Temperatur, Salinität in Meeresnähe, Holzmasse von Wäldern)
- Luftreinheit (Schadstoffkonzentrationen, NO_x, NH_3, N_2O)
- Verwaltung (zeitliche Veränderung der Bevölkerung)
- Wirtschaftsdaten (Umsätze von Filialen, Produktionsdaten von Betrieben oder einzelnen Maschinen, Monitoring von Schadstoffausstoß von Industriebetrieben)

Als Zusatzfunktion zu den punktuellen Messpunkten wurde die Option implementiert, Polygone als Datenträger in die Applikation einzuspielen. Diese können zweierlei Informationen enthalten, einen Relativwert, der als Farbe der Fläche dargestellt wird, sowie Absolutwerte, die als Pie Chart in der Mitte des jeweiligen Polygons erstellt werden. Mit dieser Funktion können unter anderem Wahlergebnisse dargestellt werden, während die dazugehörigen Polygone die Wahlbeteiligung der jeweiligen Verwaltungseinheit abbilden. Weitere Anwendungsgebiete wären Ertragsveränderungen eines Unternehmens mit absoluten Absatzzahlen nach Sektoren, oder relative Beschäftigungsquoten von Gemeinden mit absoluten Beschäftigungszahlen nach Wirtschaftssektoren.

Abbildung 28 zeigt das Wahlergebnis der Nationalratswahlen 2017 in Österreich. Die Farbe der Fläche der Bundesländer spiegelt die jeweilige Wahlbeteiligung wider, relativ zur minimalen beziehungsweise maximalen Wahlbeteiligung des jeweiligen Bundeslandes bei den letzten Nationalratswahlen. Die Kreisdiagramme stellen die anteiligen Stimmen für die drei Parteien SPÖ, ÖVP und FPÖ sowie für sonstige Parteien dar. In diesem Fall wurden die jeweiligen allgemein bekannten Parteifarben herangezogen.

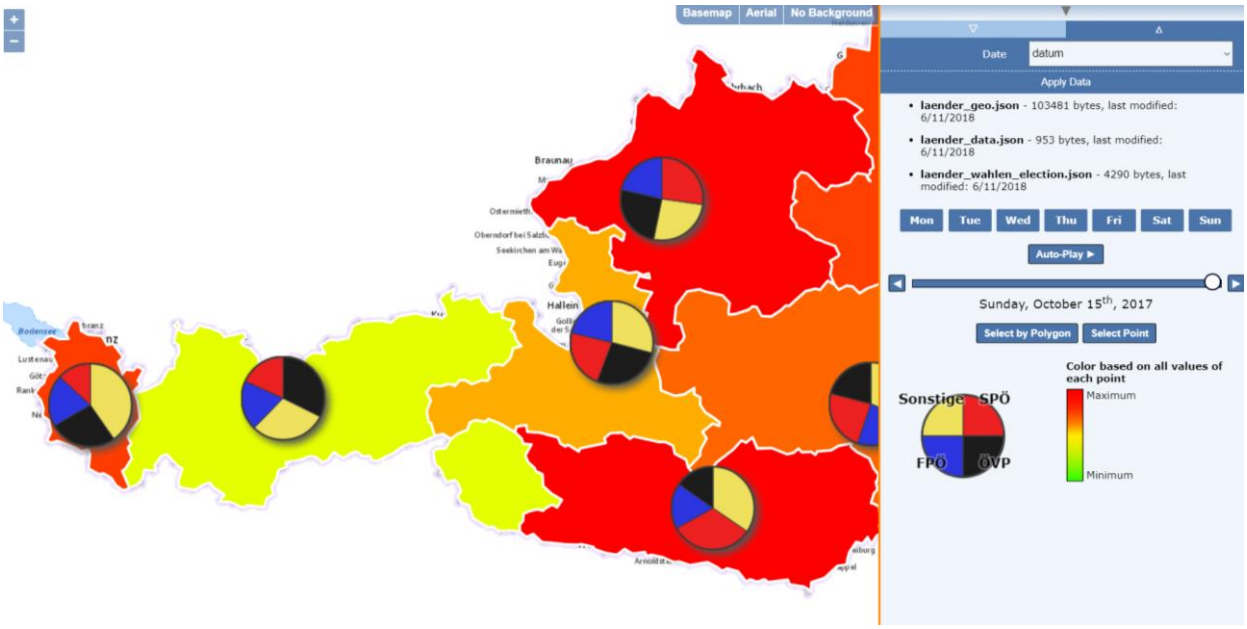


Abbildung 28 Visualisierung von Wahlergebnissen mit Mapoch (Datengrundlage: Bundesministerium für Inneres)

Die Applikation funktioniert durch die Webmercator-Projektion 3857 und durch WMTS-Grundkarten maßstabsunabhängig. Die Messpunkte können global verteilt sein oder sich innerhalb eines einzelnen Gebäudes befinden. Dadurch lassen sich auch Phänomene in einem sehr großen Maßstab visualisieren, wie zum Beispiel die Population und Honigproduktion von Bienenstöcken oder die Wahlergebnisse jedes Wahllokals einer Stadt.

In modernen Internet-Applikationen wird mit Node-Modulen gearbeitet. Node.js ist eine cross-platform Javascript runtime, basierend auf der Google Chrome V8 Engine für Server- und Desktop. In Node können ECMAScript 6-Features verwendet und über den Babel-Transpiler in ECMAScript 5 umgewandelt werden, sowie alle Sprachen die nach JavaScript transpilen, wie etwa TypeScript und CoffeeScript.[61]

Eines der Hauptfeatures von Node ist npm, der Node Package Manager. Alle geläufigen Softwarepakete sind als Node-Module über npm inkludier- und verwaltbar, so auch die in dieser Masterarbeit verwendete Software OpenLayers und Chart.js. Anders als JavaScript kann Node selbst nicht im Browser ausgeführt werden, daher müssen die verwendeten Module und JavaScript-Dateien mit einem Bundler wie Browserify, Webpack oder Rollup in für den Browser verständliches JavaScript umgewandelt werden. Plug-Ins wie Uglify oder Minify können verwendet werden, um die Größe des erstellten Bundles zu minimieren. Verwendete ECMAScript-6-Node-Module können, zum Beispiel mit Webpack, über „Tree-Shaking“ noch weiter minimiert

[61] HELLER, M. (2017): What is Node.js? The JavaScript runtime explained.

werden, indem ganze nicht-verwendete Codepassagen („dead-code“) nicht in das endgültige Bündel inkludiert werden[62].

Um die Zugänglichkeit dieser Masterarbeit zu gewährleisten und den Fokus auf das Wesentliche zu bewahren, wurde im folgenden Praxisteil dieser Arbeit vollständig auf den Einsatz von Node verzichtet. Als einzige Hilfestellung wurde das Node-Modul `http-server` zur Erstellung eines lokalen http-Servers verwendet.

Die verwendete Software OpenLayers, proj4.js und Chart.js wurde zur besseren Transparenz des Codes über Script-Tags hardcodiert in der index.html inkludiert.

Mapoch ist online zugänglich unter:
https://robertorthofer.github.io/mapoch/openlayers/

6.2 Initialisierung

Zum Programmstart wird ein globales Objekt `mapoch` erstellt (siehe Code 5). Dies verhindert eine „namespace-pollution“ durch eventuelle Drittsoftware. In diesem Objekt werden alle zukünftig notwendigen Daten und Selektionszustände zwischengespeichert, bei der Initialisierung befinden sich alle Werte auf ihrem Anfangswert.

Code 5 globales Objekt zur Vermeidung von "namespace-pollution" (Quelle: eigener Code)

```
var mapoch = {
    currentFiles: {},
    zaehlstellen_data: [],
    PieChartData: {},
    PieChartColorMap: {},
    PieChartCanvasElements: {},
    selectionStatus: {
        coords: false,
        date: false,
        pieCharts: false
    }
}
```

Um eine ausreichende Übersicht zu gewährleisten, wurde JavaScript hauptsätzlich als funktionale Programmiersprache eingesetzt, wobei die entsprechenden Funktionen als globale Funktionen deklariert und auf mehrere JavaScript-Dateien aufgeteilt wurden.

6.3 Clientseitiger Datenzugriff

Der Zugriff auf die Daten des Benutzers ist eine der grundlegenden Problemstellungen und gleichzeitig die erste vom Benutzer erforderte Aktion. Neben technischen Überlegungen stellen sich hier auch Fragen zu den verlangten Datenformaten

[62] WEBPACK (2017): Tree Shaking. https://webpack.js.org/guides/tree-shaking/

beziehungsweise Datenstruktur und deren Kommunikation mit dem Benutzer (mehr dazu im Kapitel 4.2.7: *Welche Datenformate unterstützen?*).

Die clientseitige Dateneinspeisung und -verarbeitung ist eines der Hauptfeatures der Applikation „Mapoch“. Sie bietet Vorteile bezüglich Performance, Flexibilität, Unabhängigkeit und Datensicherheit gegenüber freier und proprietärer serverseitiger Software.

6.3.1 DATENFORMAT

Der erwartete Workflow des durchschnittlichen Benutzers der Zielgruppe sieht feste Messpunkte vor, mit wechselnden beziehungsweise ständig erneuerten Messdaten. Es scheint daher sinnvoll, diese Datensätze zu trennen. Die Geo-Daten der Messpunkte sollen von Fachexperten in einer geeigneten GIS-Software schnell erstellt werden können, jedoch sollen auch für Laien diese Messpunktdaten ohne GIS-Kenntnisse problemlos manuell erstellbar sein. Der Zeitaufwand der zweiten Methode soll sich ebenso in Grenzen halten.

Unter Berücksichtigung dieser und der in Kapitel 4.2.7 besprochenen Punkte wurden als unterstützte Datenformate die offenen und weit verbreiteten Standards CSV und JSON beziehungsweise GeoJSON gewählt. CSV beschreibt lediglich ein Dateiformat, für ein problemloses Einlesen der Daten müssen dem Benutzer zusätzliche Informationen geliefert werden.

6.3.1.1 Dateiformate der Messpunkte

GeoJSON

GeoJSON ist ein weit verbreitetes Austauschformat geographischer Daten (vgl. Kapitel 4.2.2: *GeoJSON*). Es ist mit jedem geläufigen Desktop-GIS generier- und editierbar. Da es sich um ein lesbares Datenformat in Textform handelt, kann es theoretisch in jedem Texteditor erstellt werden, jedoch kann diese Art der Datenerstellung für größere Datensätze, insbesondere für Laien, sehr mühsam sein. Alternativ existieren zur Erstellung von GeoJSON auch online-Tools, wie etwa geojson.io[63]. In solchen Programmen können einfache Geometrien direkt im Browser erstellt und attributiert werden, das fertige GeoSJON kann dann entweder heruntergeladen oder direkt aus dem integrierten Editor kopiert werden.

[63] MAPBOX (2017): geojson.io. https://github.com/mapbox/geojson.io

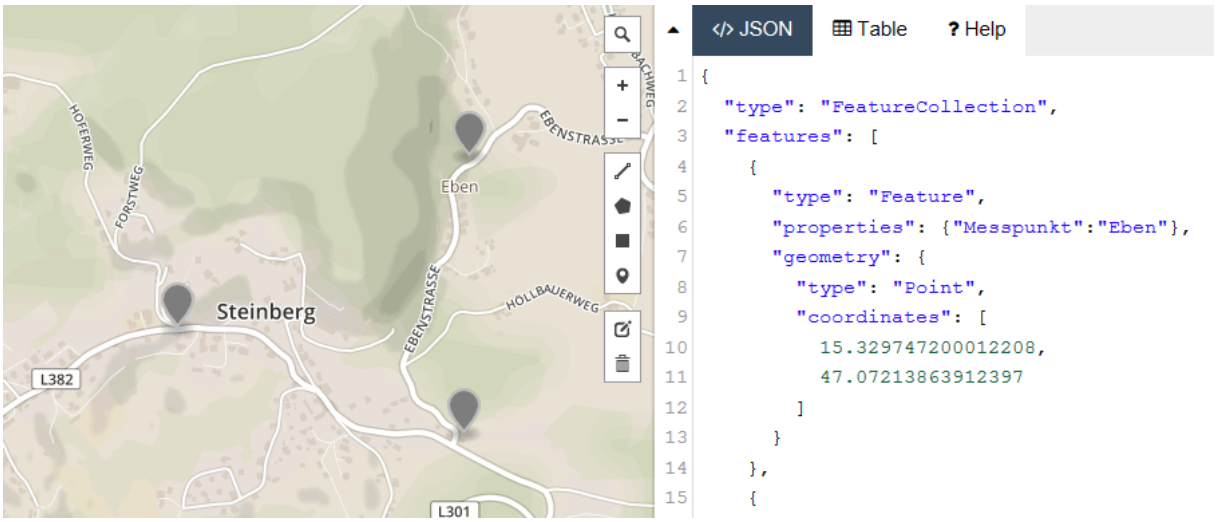


Abbildung 29 Erstellung eines GeoJSON im Browser über geojson.io von Mapbox (Quelle: eigener Code in geojson.io)

Das geforderte GeoJSON muss eine FeatureCollection mit einem Array aus Features darstellen, wobei jedes Feature im Attribut „`properties`“ ein `object` mit einem eindeutigen Namen des Messpunktes als `string` aufweisen muss (siehe Code 6). Weitere `properties` sind erlaubt, werden jedoch nicht weiterverarbeitet.

Code 6 Beispielhaftes GeoJSON der Messpunkte (Quelle: eigener Code)

```
{"type":"FeatureCollection",
  "features":[
    {
      "type":"Feature",
      "geometry":{
        "type":"Point",
        "coordinates":[16.0930173999721,47.0503057773741]
      },
      "properties":{"zaehlstelle":"b31901"}
    },
    {
     …
  ]
}
```

CSV

CSV wird zusätzlich zu jedem Tabellenkalkulationsprogramm von einer Vielzahl von Software unterstützt. Ein großer Vorteil von CSV ist, dass durch die weite Verbreitung von Excel und ähnlichen Programmen jeder Computerbenützer diese Datensätze erstellen beziehungsweise editieren kann.

Dadurch, dass der GeoJSON-Standard ausschließlich das Projektionssystem EPSG:4326 unterstützt, können Daten anderer Projektionssysteme, wie etwa lokaler oder regionaler Systeme, über CSV importiert werden. Der EPSG-Code dieses Projektionssystems muss jedoch dem Benutzer bekannt sein.

Das für Mapoch entwickelte CSV-Format ist sehr offen. Verlangt werden drei Spalten, eine Spalte für die Messpunktnamen, zwei Spalten für das Koordinatenpaar.

Zusätzlich wird ein `header` verlangt, in diesem stehen die Spaltennamen. Getrennt werden die Werte durch Beistrich oder Strichpunkt, das Komma der Koordinaten muss ein Punkt sein. Die Reihenfolge der Spalten wird nicht vorgegeben.

Code 7 Beispielhaftes CSV der Messpunktkoordinaten im Projektionssystem EPSG:31259 (Quelle: eigener Code)

```
zaehlstelle,x_coord,y_coord
b31901,731824.0443,212378.5118
b11401,610255.2514,240538.9853
b32001,547499.2024,253401.9068
b05425,724181.4392,243783.2969
```

6.3.1.2 Dateiformate der Messwerte

Analog zu den Datenformaten der Lage der Messpunkte werden auch für die dazustellenden Werte JSON und CSV unterstützt. Wichtig ist dabei, dass die Namen der Messpunkte übereinstimmen, damit eine Datensynthese möglich wird. Die Namen müssen dabei auch bezüglich Groß- und Kleinschreibung übereinstimmen, um jegliche Ambiguität zu vermeiden.

JSON

Das JSON-Format, auf dem auch GeoJSON basiert, ist ein weit verbreitetes Datenaustauschformat im Web. In Mapoch wird für jede Zeiteinheit ein `object` verlangt, mit je einem Wert für jeden Messpunkt. Der key zum `date-string` muss „`datum`“ sein, bei den Datenpaaren muss der `key` den Namen des Messpunktes repräsentieren, der `value` ist der dazugehörige Messwert (siehe Code 8).

Code 8 Beispielhaftes JSON der Messdaten (Quelle: eigener Code)

```
[
  {
   "datum":"2014-01-01",
   "b02501":704,
   "b06404":3818,
   "b06603":2542
  },
  {
    "datum":"2014-01-02",
    "b02501":1055,
    "b06404":8963,
    "b06603":6297
  },
  {
    "datum":"2014-01-03",
    "b02501":1023,
    "b06404":9243,
    "b06603":6023
  }
]
```

CSV

Das Dateiformat der Messwerte muss besonders zugänglich für den Benutzer sein. Oft wird eine Vielzahl von Messwerten in tabellarischer Form verwaltet. CSV ist in Tabellenkalkulationsprogrammen für jeden zugänglich und schnell generierbar.

Code 9 zeigt ein unterstütztes CSV der Messdaten. Die äußerst linke Spalte enthält die Verknüpfungs-IDs, die erste Reihe (Header-Reihe) enthält die Zeitpunkte.

Code 9 Beispielhaftes CSV der Messdaten (Quelle: eigener Code)

```
datum,2014-01-01,2014-01-02,2014-01-03,2014-01-04
b02501,704,1055,1023,953
b06404,3818,8963,9243,7600
b06603,2542,6297,6023,4916
```

Datumsformat

Das Datumsformat (beziehungsweise das Zeitformat) spielt unabhängig des gewählten Dateiformates eine essentielle Rolle. In Mapoch wird das vom W3C empfohlene ISO 8601[64] unterstützt, der internationale Standard für die Repräsentation von Datum und Zeit.

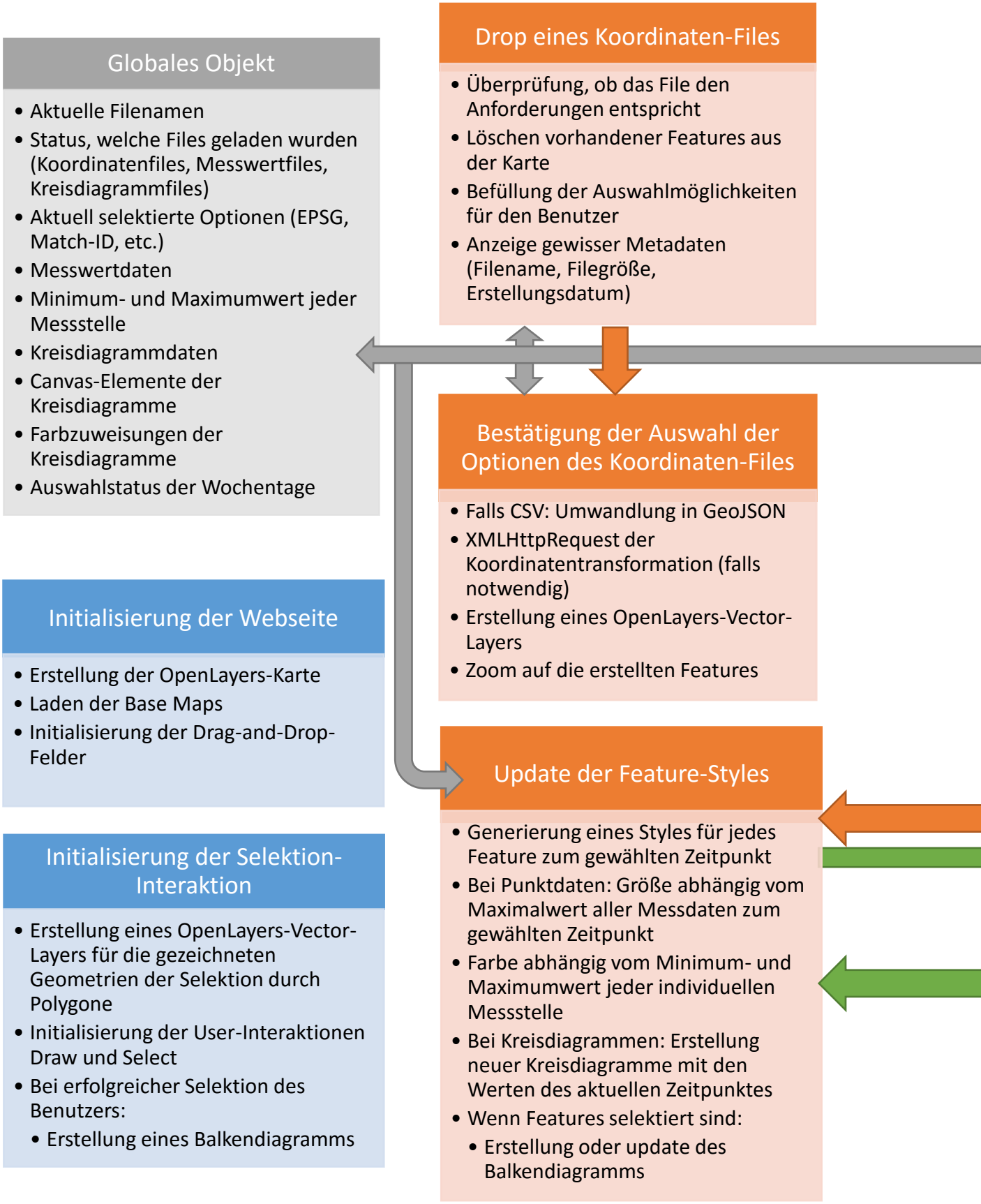
Dieses Format wird von den meisten Tabellenkalkulationsprogrammen und von anderer Software mit Zeitbezug unterstützt. Das Format für einen Tag plus Uhrzeit und Zeitzone lautet `YYYY-MM-DDThh:mm:ss.sTZD`, wobei der string laut dem Standard von hinten verkürzt werden kann. So lautet das Format für einen Tag `YYYY-MM-DD`, für ein Monat `YYYY-MM`.

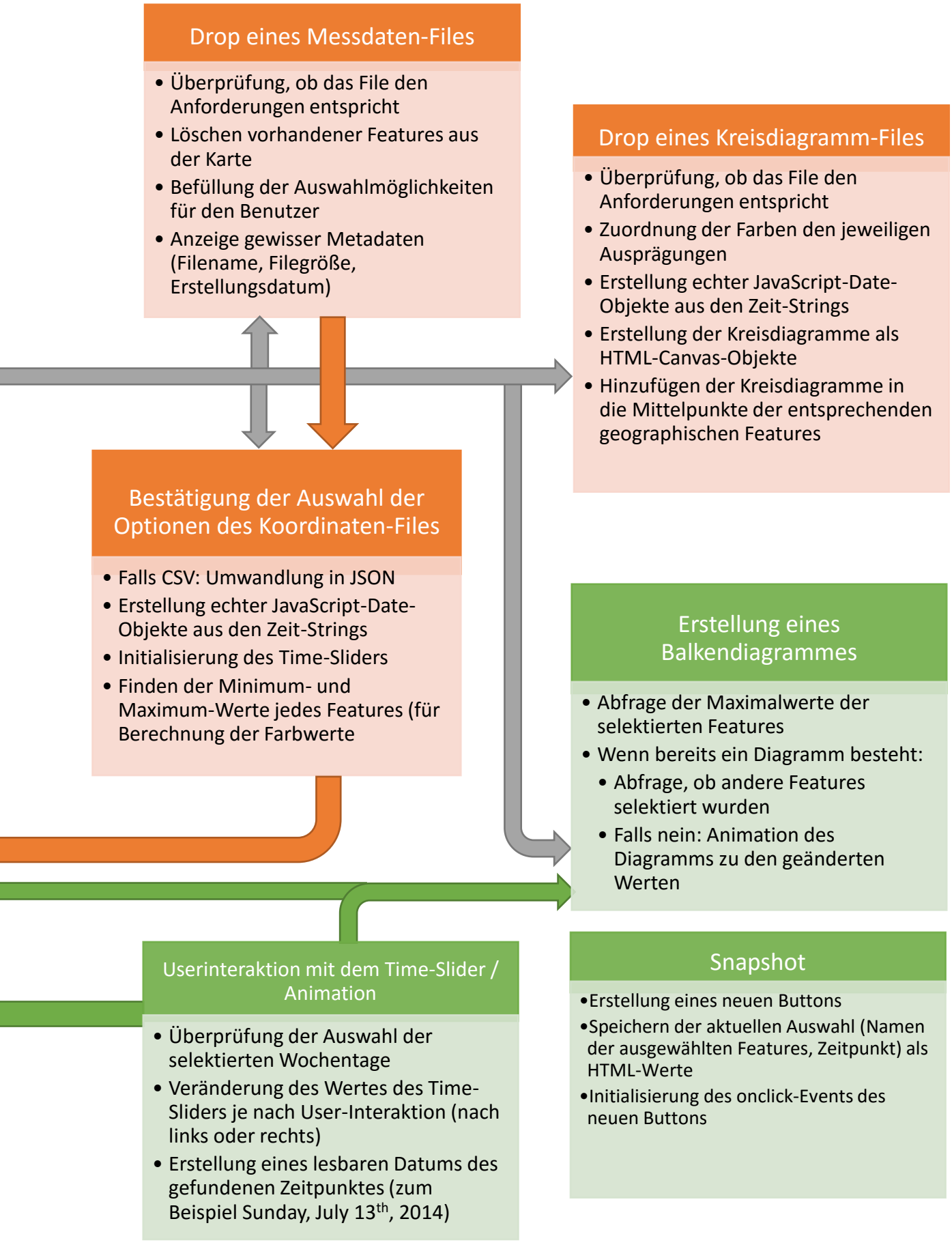
Mapoch unterstützt durch die Verwendung des JavaScript-Standardobjectes `date` alle Zeiteinheiten des ISO 8601-Formates[65], wobei der volle Funktionsumfang nur für Tagesdaten verfügbar ist. Dies äußert sich unter anderem durch das Feature, in dem der Benutzer einzelne Wochentage ein- beziehungsweise ausschalten kann, um so zum Beispiel eine Animation aller Sonntage in seinen Daten zu erhalten.

Liegen etwa jährliche, monatliche oder stündliche Einheiten vor, müssten kleinere Anpassungen durchgeführt werden. Bei stündlichen Daten wäre zum Beispiel eine Uhr als Zeitlegende sinnvoll, bei monatlichen Daten wäre ein Kalender oder eine radiale Legende für diese Aufgabe besser geeignet.

[64] DUBOST, K. (2016): Use international date format (ISO). http://www.w3.org/QA/Tips/iso-date
[65] MOZILLA AND INDIVIDUAL CONTRIBUTORS (2018): date https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

6.3.2 ÜBERBLICKSDIAGRAMM MAPOCH




Drop eines Messdaten-Files
• Überprüfung, ob das File den Anforderungen entspricht
• Löschen vorhandener Features aus der Karte
• Befüllung der Auswahlmöglichkeiten für den Benutzer
• Anzeige gewisser Metadaten (Filename, Filegröße, Erstellungsdatum)
Drop eines Kreisdiagramm-Files
• Überprüfung, ob das File den Anforderungen entspricht
• Zuordnung der Farben den jeweiligen Ausprägungen
• Erstellung echter JavaScript-Date-Objekte aus den Zeit-Strings
• Erstellung der Kreisdiagramme als HTML-Canvas-Objekte
• Hinzufügen der Kreisdiagramme in die Mittelpunkte der entsprechenden geographischen Features
Bestätigung der Auswahl der Optionen des Koordinaten-Files
• Falls CSV: Umwandlung in JSON
• Erstellung echter JavaScript-Date-Objekte aus den Zeit-Strings
• Initialisierung des Time-Sliders
• Finden der Minimum- und Maximum-Werte jedes Features (für Berechnung der Farbwerte
Erstellung eines Balkendiagrammes
• Abfrage der Maximalwerte der selektierten Features
• Wenn bereits ein Diagramm besteht:
• Abfrage, ob andere Features selektiert wurden
• Falls nein: Animation des Diagramms zu den geänderten Werten
Userinteraktion mit dem Time-Slider / Animation
• Überprüfung der Auswahl der selektierten Wochentage
• Veränderung des Wertes des Time-Sliders je nach User-Interaktion (nach links oder rechts)
• Erstellung eines lesbaren Datums des gefundenen Zeitpunktes (zum Beispiel Sunday, July 13th, 2014)
Snapshot
•Erstellung eines neuen Buttons
•Speichern der aktuellen Auswahl (Namen der ausgewählten Features, Zeitpunkt) als HTML-Werte
•Initialisierung des onclick-Events des neuen Buttons

6.3.3 DRAG&DROP

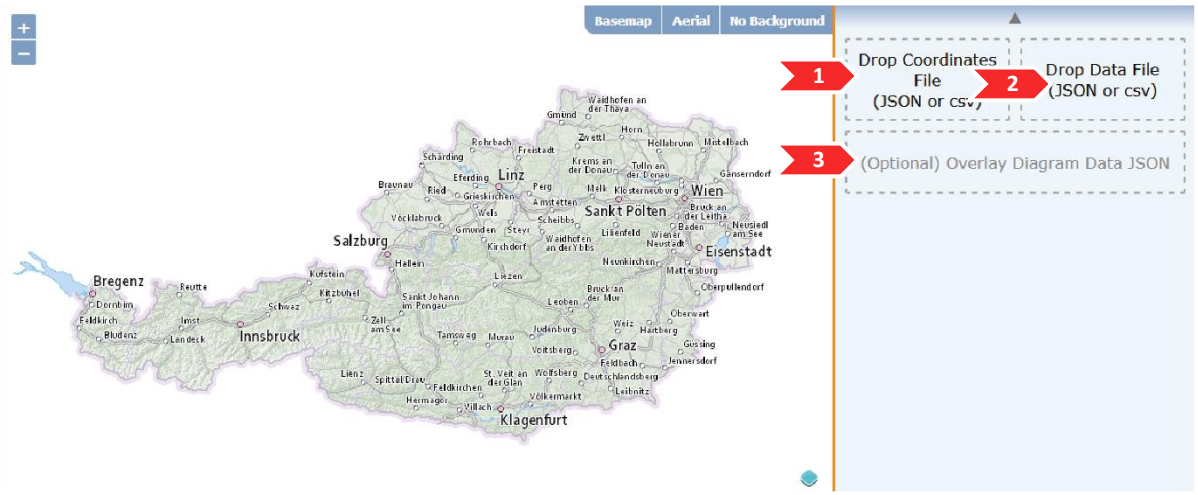


Abbildung 30 1: Drop-Zone des Koordinaten-Files, 2: Drop-Zone des dazugehörigen Daten-Files, 3: Drop-Zone für optionale Pie Chart Daten (Quelle: eigene Abbildung)

Der Datenimport in die Applikation erfolgt über die HTML5 Drag & Drop API. Mit ihr können Dateien im Browser asynchron importiert, über einen `FileReader` gelesen und anschließend verarbeitet werden. Im Unterschied zur FileSystem API werden alle notwendigen Funktionen der Drag & Drop API von sämtlichen gängigen Browsern unterstützt[66]. Ein weiterer Vorteil ist eine mögliche niedrigere Hemmschwelle beim Einlesen sensibler Daten über Drag&Drop im Vergleich zu einem klassischen FileSystem-Dialog, da der Benutzer diesen leichter mit dem Upload auf einen Server in Verbindung bringen könnte.

6.3.4 INITIALISIERUNG DER DROP-ZONES

Um mit Drag&Drop Dateien einlesen zu können werden „Drop-Zones“ benötigt, also `div`-Elemente, die bei einem `dragover`- und bei einem `drop`-Event über `callback-functions` die Dateien entsprechend weiterreichen. Code 10 zeigt die Initialisierung dieser Bereiche in Mapoch. Es werden drei `div` der HTML als Drop-Zones aufbereitet, eine für die Messpunktdatei, eine für die Messwertedatei und eine für optionale Kreisdiagramme.

Code 10 Initialisierung der drop zones (Quelle: eigener Code)

```
function init_dropzone() {
    var dropZone1 = document.getElementById('drop_zone_data_file');
    dropZone1.addEventListener('dragover', handleDragOver, false);
    dropZone1.addEventListener('drop', handleCoordsFile, false);

    var dropZone2 = document.getElementById('drop_zone_coords_file');
```

[66] MOZILLA AND INDIVIDUAL CONTRIBUTORS (2018): FileSystem https://developer.mozilla.org/en-US/docs/Web/API/FileSystem

```
    dropZone2.addEventListener('dragover', handleDragOver, false);
    dropZone2.addEventListener('drop', handleDataFile, false);

    var dropZone3 = document.getElementById('optional_drop_zone');
    dropZone3.addEventListener('dragover', handleDragOver, false);
    dropZone3.addEventListener('drop', handlePiechartFile, false);
}
```

Das `dragover`-Event wird aktiviert solange das Element vom Benutzer über dem gewählten `div` gehalten wird. Standardmäßig passiert nichts, wenn ein Element in den Browser gezogen oder im Browser ausgelassen wird. Dieses standardmäßige Verhalten wird mit `evt.stopPropagation()` und `evt.preventDefault()` unterbunden (siehe Code 11). Weiters kann über `evt.dataTransfer.dropEffect = 'copy'` dem Benutzer visuell mitgeteilt werden, dass er sich mit dem gezogenen Element nun über einer Drop-Zone befindet und dass das Element fallen gelassen werden kann. Der Browser verändert dadurch bei einem `dragover`-Event entweder den Cursor oder verändert das Aussehen des gezogenen Elements.

Code 11 dragover-Event callback function der Drop-Zones (Quelle: eigener Code)

```
function handleDragOver(evt) {
    evt.stopPropagation();
    evt.preventDefault();
    evt.dataTransfer.dropEffect = 'copy'; // show Effekt at cursor when dragging
}
```

6.3.5 HANDLING DER GEDROPPTEN DATEIEN

Das Einlesen der Daten unterscheidet sich für jede der zwei beziehungsweise drei geforderten Dateien. In diesem Teil wird exemplarisch der Einlesevorgang der Messpunktdaten dargestellt.

Die `dragover`-Event `callback function` ist für alle Drop-Zones dieselbe. Im Unterschied dazu muss für jede Drop-Zone eine eigene `drop`-Event `callback function` definiert werden, da die unterschiedlichen Dateien unterschiedlich weiterverarbeitet werden. Der `drop`-Event wird dann ausgelöst, wenn der Benutzer das Element über der definierten Drop-Zone auslässt. Auch hier muss die Standardfunktion des Browsers mit `evt.stopPropagation()` beziehungsweise `evt.preventDefault()` unterdrückt werden, da der Browser sonst versuchen würde, die Datei zu öffnen, sollte es eine Textdatei sein. Wird diese Funktion erfolgreich übergangen, kann mit `evt.dataTransfer.files` auf die gezogenen Dateien zugegriffen werden (siehe Code 12).

Code 12 drop-Event callback function der Drop-Zones der Koordinaten-Datei (1) (Quelle: eigener Code)

```
function handleCoordsFile(evt) {
    evt.stopPropagation();
    evt.preventDefault();
    var files = evt.dataTransfer.files;
[…]
```

6.3.6 EINLESEN DER MESSPUNKTGEOMETRIEN

Ist der Datentransfer des Browsers abgeschlossen, kann auf das `filelist`-Objekt zugegriffen werden. Dieses beinhaltet die Daten aller gezogenen Dateien als `array`. Wurde kein File, sondern Text oder ein anderes `dragable` Element in die Drop-Zone gezogen, wird diese zwar aktiviert, die Länge des `filelist`-Objektes ist in diesem Fall jedoch 0. Wurde keine Datei oder zu viele Dateien in die Drop-Zone gezogen, wird dies dem Benutzer entsprechend kommuniziert und die Funktion abgebrochen (siehe Code 13).

Code 13 drop-Event callback function der Drop-Zones der Koordinaten-Datei (2): Überprüfung der Anzahl der Dateien (Quelle: eigener Code)

```
[…]
if ((files.length > 1) || (files.length == 0)) {
    alert("Please drop only a single file");
    return;
}
[…]
```

Wurde korrekterweise ein einzelnes File geladen, kann anschließend der Media-Type (MIME) der Datei über `file.type` überprüft werden. Als Messpunkte sind die Typen `application/json` und `text/csv` erlaubt. GeoJSON entspricht dem JSON-Standard und wird somit mit diesem MIME-Type assoziiert. Leider treten in den Browsern Internet Explorer 10/11 und Microsoft Edge unregelmäßig Probleme mit dem Attribut `type` der geladenen Datei auf, so dass dieses Feld teilweise einen leeren String enthält. Daher muss, speziell für diese Browser, eine weitere Abfrage über die Dateiendung erfolgen (siehe Code 14). Hierzu werden vom Dateinamen die letzten drei beziehungsweise vier Zeichen entnommen und überprüft.

Entspricht die geladene Datei nicht einem der erlaubten Dateitypen, wird dies dem Benutzer mitgeteilt und die Funktion abgebrochen.

Code 14 drop-Event callback function der Drop-Zones der Koordinaten-Datei (3): Überprüfung des Dateityps (Quelle: eigener Code)

```
[…]
var file = files[0];
var isCSV = false;
var isJSON = false;
if (file) {
  console.log("type of file: " + file.type);
  if (file.type === "application/json" ||
  file.name.substr(file.name.length  - 4) === "json") {
      isJSON = true;
  }
  if (file.type === "text/csv" || file.name.substr(file.name.length - 3) ===
  "csv") {
    isCSV = true;
  }

  if (!isCSV && !isJSON) {
```

```
    alert("Please make sure you drop files of the allowed type\n(your type:
    " + files[0].type + "). \n\n Allowed file types are JSON and CSV");
    return;
  }
}
[…]
```

Ist der Datentyp erlaubt, wird überprüft, ob im global definierten Objekt `mapoch` (siehe Kapitel 6.2: *Initialisierung*) der Wert für `currentFiles.Coords` undefiniert ist. Ist er es nicht, bedeutet dies, dass der Benutzer bereits eine Messpunktdatei eingelesen hat. Dies können entweder alte Messpunkte sein, die vom Benutzer nicht weiter gebraucht werden, oder der Benutzer hat die neue Datei versehentlich in die Drop-Zone geladen.

Mit der Funktion `confirm()` wird der Benutzer gefragt, ob die alten Daten verworfen werden und die neuen eingelesen werden sollen (siehe Code 15). Bejaht der Benutzer dies, werden die alten Optionen zur Selektion von X-Koordinate, Y-Koordinate und Name des Messpunktes gelöscht und der Datenimport wird fortgesetzt. Andernfalls werden die neuen Daten verworfen, die alten bleiben bestehen.

Die `select`-Elemente können entweder komplett gelöscht und neu erstellt werden, oder es werden alle Optionen entfernt. Im zweiten Fall wird aufgrund der sich veränderten Länge der Auswahlmöglichkeiten immer die erste Option gelöscht, und zwar so oft, wie Optionen am Anfang der Schleife vorhanden waren (`select.options.length`).

Code 15 drop-Event callback function der Drop-Zones der Koordinaten-Datei (4): Überschreiben vorhandener Daten (Quelle: eigener Code)

```
[…]

if (typeof(mapoch.currentFiles.Coords) !== "undefined") {
  var r = confirm("Override existing File?"); // ask User
  if (r == true) {
    console.log("Override File");
    // now clear all old options from Coords- and Match-ID-Selection
    var select = document.getElementById("coordIDSelect");
    var select2 = document.getElementById("xSelect");
    var select3 = document.getElementById("ySelect");
    var length = select.options.length;
    for (i = 0; i < length; i++) {
      select.options[0] = null;
      select2.options[0] = null;
      select3.options[0] = null;
    }
  } else {
    //Do nothing
    return;
  }
}

[…]
```

Zum jetzigen Zeitpunkt wurde das File soweit eingelesen, dass die Metadaten wie Dateiname und Dateityp verfügbar sind. Die im File vorhandenen Daten müssen erst mit einem `FileReader` gelesen werden, um auf sie zugreifen zu können.

Für den `FileReader` kann eine `onload`-Function definiert werden, welche aufgerufen wird, wenn die Datei fertig gelesen wurde. Darin kann auf den eingelesenen Inhalt der Datei über `FileReader.result` zugegriffen werden. Diese `onload`-Function wird definiert, bevor der `FileReader` die Datei mit `reader.readAsText()` einliest.

Code 16 zeigt die unterschiedliche Verarbeitung der Daten im FileReader. Ist die Datei vom Typ csv, so werden Auswahlmöglichkeiten (`columnNames`) für die Spalte der Match-ID sowie X- und Y-Koordinate der Messpunkte bereitgestellt (siehe Kapitel 6.4.2: *Erstellen des Auswahl-Menüs für den Benutzer (CSV)*). Dieser Schritt ist bei GeoJSON nicht notwendig, die Geometrie ist in diesem Format standardisiert.

Code 16 Einlesen der Koordinaten-Datei mit einem FileReader. Die Datei wird je nach Datentyp (GeoJSON oder csv) unterschiedlich weiterverarbeitet. (Quelle: eigener Code)

```
[…]
coords_json = {};
var reader = new FileReader(); // to read the FileList object
reader.onload = function(event) {
    var columnNames = [];
    if (isCSV) { // check if filetiype is csv
        mapoch.currentFiles.CoordsFileType = "csv";
        columnNames = getColumnNames(reader.result);
        window.csv = reader.result; // temporary save into global
        populateSelection(columnNames, 1);
    } else if (isJSON) {
        mapoch.currentFiles.CoordsFileType = "JSON";
        coords_json = JSON.parse(reader.result);
        populateSelection(columnNames, 1);
    } else {
        alert("Unrecognized Filetype. Please Check your input (only .csv or .json
        allowed)");
    }

    document.getElementById("hideCoordSelection").style.visibility = "visible";
    document.getElementById("choseFieldDiv1").style.visibility = "visible";
    document.getElementById("renderCoordinatesButton").style.visibility = "visible";
    document.getElementById("hideSelectionHolder").style.visibility = "visible";

    document.getElementById("renderCoordinatesButton").addEventListener('click',
    function() {
        add_zaehlstellen(coords_json);  // conversion into native JS later
    }, false);
};
reader.readAsText(file, "UTF-8");
```

6.4 Benutzerauswahl der benötigten Attribute

6.4.1 ERSTELLEN DES AUSWAHL-MENÜS FÜR DEN BENUTZER (GEOJSON)

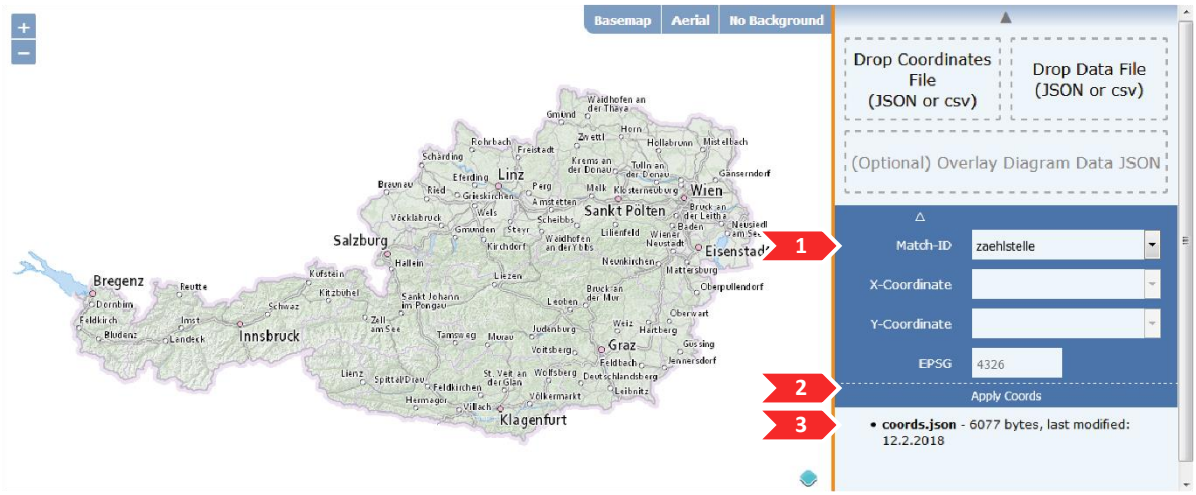


Abbildung 31 Mapoch nach Drag&Drop einer GeoJSON-Koordinatendatei, 1: Menü zur Auswahl der Match-ID zu den dazugehörigen Daten (X-Koordinate, Y-Korrdinate und EPSG-Code sind nur für csv erforderlich), 2: Button zum Bestätigen der Auswahl, 3: Metadaten der eingelesenen Datei (Quelle: eigene Abbildung)

Ist das eingelesene Koordinatenfile ein GeoJSON, so muss der Benutzer die Match-ID angeben. Die Werte müssen im Koordinatenfile und im Messdatenfile übereinstimmen, damit eine erfolgreiche Verknüpfung der Daten möglich ist.

Um die Benutzerfreundlichkeit aufrecht zu erhalten, wird nach erfolgreichem Einlesen überprüft, ob die Datei das `property` “features“ enthält und, falls dem so ist, ob geographische Features vorhanden sind (siehe Code 17). Es ist jedoch nicht möglich zu überprüfen, ob die Koordinaten korrekt sind oder ob sich die Features im richtigen Koordinatensystem des GeoJSON-Standards (EPSG: 4326) befinden.

Code 17 Check ob ein korrektes GeoJSON geladen wurde (Quelle: eigener Code)

```
if(!coords_json.hasOwnProperty("features")){
  alert("Please check if the dropped file is a correct GeoJSON");
  return;
}
if(coords_json.features.length == 0){
  alert("Please check if the dropped file has geographic features");
  return;
}
```

Weitere Informationen befinden sich im `properties`-Attribut der einzelnen Features. Es wird davon ausgegangen, dass jedes Feature über dieselben `properties` verfügt, Nullwerte sind jedoch erlaubt. Aus diesem Grund werden für die Benutzerauswahl der Match-ID alle `propertyNames` des ersten Features gelesen und in einem drop-down-Menü (`HTML select element`) als mögliche Optionen hinzugefügt (siehe Code 18). Die

Auswahl der Koordinatenfelder wird deaktiviert, da der GeoJSON-Standard nur Koordinaten des EPSG 4326 unterstützt.

Code 18 Lesen der propertyNames und Hinzufügen zum vorhandenen HTML select element "coordIDSelection" (Quelle: eigener Code)

```
var propertyNames =
Object.getOwnPropertyNames(coords_json.features[0].properties) // array of
properties of GeoJSON

var coordIDSelection = document.getElementById('coordIDSelect');
// clear all existing options
coordIDSelection.options.length = 0;
var propertyNamesLength = propertyNames.length;
for (i = 0; i < propertyNamesLength; i++) {
  var opt = document.createElement("option");
  opt.value = propertyNames[i];
  opt.innerHTML = propertyNames[i];
  coordIDSelection.appendChild(opt);
}
document.getElementById("xSelect").disabled = true;
document.getElementById("ySelect").disabled = true;
```

6.4.2 ERSTELLEN DES AUSWAHL-MENÜS FÜR DEN BENUTZER (CSV)

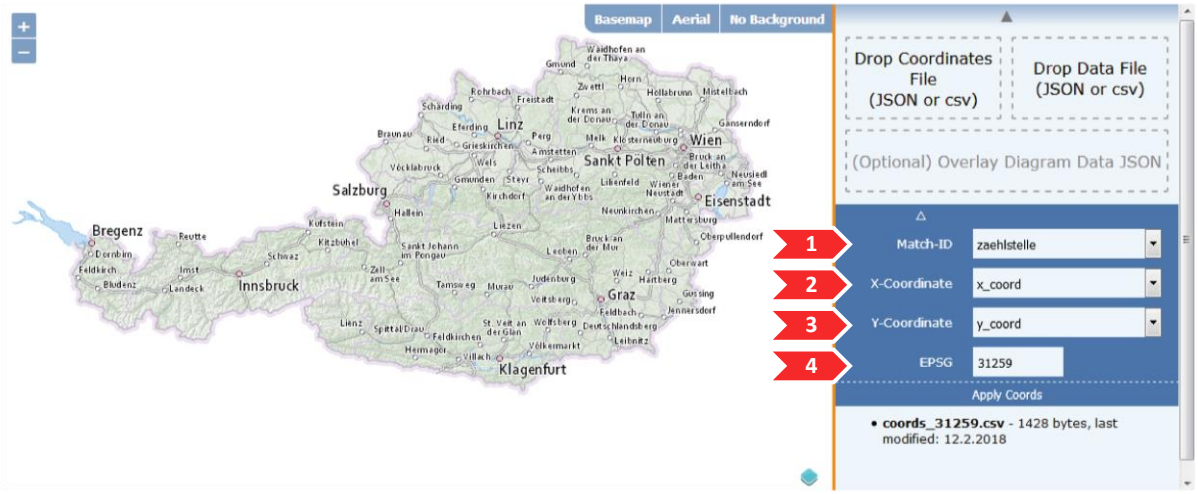


Abbildung 32 Mapoch nach Drag&Drop einer csv-Koordinatendatei, 1: Menü zur Auswahl der Match-ID zu den dazugehörigen Daten, 2: Auswahl der X-Koordinaten, 3: Auswahl der Y-Korrdinaten, 4: Eingabe des EPSG-Codes (Quelle: eigene Abbildung)

Da keine Standardisierung für geographische Daten als csv-Format existiert und die Applikation Mapoch möglichst zugänglich für verschiedene Benutzergruppen sein soll, ist ein Auswählen der korrekten Spalten für die Match-ID, X- und Y-Koordinaten notwendig. Zusätzlich werden alle standardisierten Koordinatensysteme unterstützt, für die Transformation in das Web-Mercator Koordinatensystem wird der EPSG-Code benötigt.

Die für die Auswahl benötigten Spaltennamen `columnNames` wurden bereits beim Einlesen der Daten extrahiert und werden hier als `array` von `strings` bereitgestellt. Nach

dem Löschen etwaiger alter Auswahlmöglichkeiten werden diese Spaltennamen als `options` den dafür angedachten `HTML select elements` hinzugefügt (siehe Code 19).

Code 19 Befüllen der Auswahl der Match-ID, der X- und der Y-Koordinate eines csv-Koordinatenfiles (Quelle: eigener Code)

```
var coordIDSelection = document.getElementById('coordIDSelect');
var coordXSelection = document.getElementById('xSelect');
var coordYSelection = document.getElementById('ySelect');

// clear all existing options
coordIDSelection.options.length = 0;
coordXSelection.options.length = 0;
coordYSelection.options.length = 0;

coordXSelection.disabled = false; // enable x- and y-coordinate Selection
coordYSelection.disabled = false;

var headerLength = columnNames.length;
for (i = 0; i < headerLength; i++) {
  var opt = document.createElement("option");
  opt.value = columnNames[i];
  opt.innerHTML = columnNames[i];
  coordIDSelection.appendChild(opt);
  var opt2 = opt.cloneNode(true); // clone Options
  coordXSelection.appendChild(opt2);
  var opt3 = opt.cloneNode(true); // clone Options
  coordYSelection.appendChild(opt3);
}
```

6.4.3 TRANSFORMATION EINER CSV-DATEI IN GEOJSON

Um geographische Features aus einer nicht standardisierten Quelle in einer OpenLayers Karte darzustellen, existieren zweierlei Möglichkeiten: entweder werden die Daten in ein standardisiertes Format transformiert, um sie anschließend mit den von OpenLayers zur Verfügung gestellten Methoden einzulesen, oder die einzelnen Features werden mit den Koordinaten und Attributen direkt als native OpenLayers-Features erstellt. In der Applikation Mapoch wurde der erste Weg gewählt, die Koordinatendatei als csv wird intern temporär in ein GeoJSON umgewandelt. Dies erhöht die Wiederverwendbarkeit der anderen Funktionen und trägt zur Wartbarkeit der Software bei.

Die Umwandlung des eingelesenen csv in ein GeoJSON kann erst nach dem Auswählen der korrekten Spaltennamen und der Bestätigung der Auswahl durch den „Apply Coords“-Button (siehe Abbildung 32) erfolgen. Der zwischengespeicherte Inhalt der csv-Datei muss in einzelne Zeilen zerlegt werden, wobei jede Zeile einem geographischen Feature entspricht. Hierbei ist zu beachten, dass viele Programme der Betriebssysteme Windows und Mac verschiedene Zeichen (`line feed` beziehungsweise `carriage return`) als Indikatoren für einen Zeilenumbruch benützen[67]. Anschließend

[67] FITZJOHN, R. (2013): Excel and line endings https://nicercode.github.io/blog/2013-04-30-excel-and-line-endings/

werden die aktuellen Werte der `HTML select elements` ausgelesen, in denen der Benutzer die Spalten für Match-ID, X- und Y-Koordinate ausgewählt hat. Aus diesen werden die korrelierenden Positionen der Werte herausgelesen. Befindet sich zum Beispiel die X-Koordinate im Header an der vierten Stelle, so befinden sich auch alle Werte der X-Koordinate an der vierten Stelle der jeweiligen Zeile (siehe Code 20).

Code 20 Erstellen eines GeoJSON aus csv (1): Trennen des Textes in einzelne Zeilen, Auslesen der Benutzereingabe und Finden der entsprechenden Spalten (Quelle: eigener Code)

```
function csvToGeoJSON() {
  var lines = csv.split(/[\r\n]+/); // split for windows and mac csv
  var headers = lines[0].split(",");
  var matchID = document.getElementById("coordIDSelect").value;
  var xColumn = document.getElementById("xSelect").value;
  var yColumn = document.getElementById("ySelect").value;

  // get the positions of the seleted columns in the header
  var positionMatchID = headers.indexOf(matchID);
  var positionX = headers.indexOf(xColumn);
  var positionY = headers.indexOf(yColumn);
[…]
```

Für jede Zeile, also jedes Feature, wird ein Objekt im GeoJSON-Format erstellt. Die Match-Id wird als `property` in jedem Feature gespeichert. Abschließend wird aus dem `array` der einzelnen Features eine GeoJSON `FeatureCollection` erstellt. Diese kann von OpenLayers importiert und in native OpenLayers-Features umgewandelt werden (siehe Code 21).

Code 21 Erstellen eines GeoJSON aus einer csv-Datei über bekannte Spaltennamen der X- und Y-Koordinate (Quelle: eigener Code)

```
[…]
 var obj_array = []
  for (var i = 1; i < lines.length - 1; i++) {
    var json_obj = {
      "type": "Feature"
    };
    var currentline = lines[i].split(",");
    json_obj["geometry"] = {
      "type": "Point",
      "coordinates": [parseFloat(currentline[positionX]),
      parseFloat(currentline[positionY])]

    };
    json_obj["properties"] = {};

    json_obj["properties"][matchID] = currentline[positionMatchID]; // get the
    name of measurement points, has to match with data file
    obj_array.push(json_obj);
  };

  var complete_geojson = {
    "type": "FeatureCollection",
    "features": obj_array // all objects of the csv
  }
  return complete_geojson; //return geoJSON
}
```

6.4.4 HINZUFÜGEN DER MESSPUNKTKOORDINATEN ZUR KARTE UND KOORDINATENTRANSFORMATION

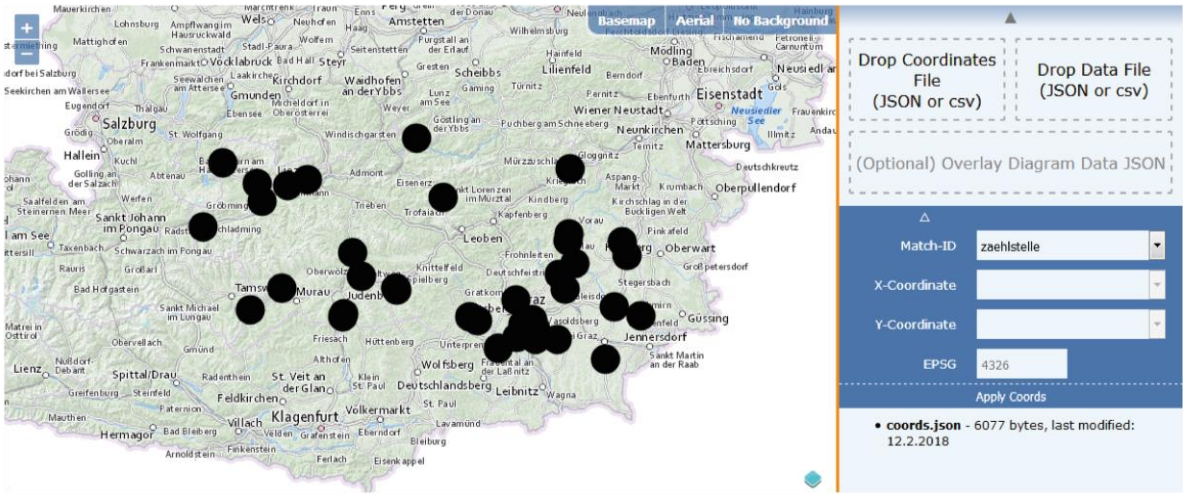


Abbildung 33 Menü nach erfolgreichem Einlesen und transformieren der Koordinaten-Datei. Die Auswahl der Spalte der X- und Y-Koordinate ist nur bei csv-Dateien notwendig. Nach erfolgreichem Einlesen wird auf die Features gezoomt. (Quelle: eigene Abbildung)

Nachdem trotz verschiedener unterstützter Datenformate die Messpunktgeometrien nach Bestätigung durch den „Apply Coords“-Button intern immer als GeoJSON vorliegen, können diese nun mit einer einzigen Funktion in der OpenLayer-Karte dargestellt werden. Zuerst werden alle alten Vector Layer, falls vorhanden, entfernt und die Benutzerauswahl wird über DOM-Abfragen in lokale Variablen zwischengespeichert (siehe Code 22).

Code 22 Vorbereitungen für die Erstellung des neuen Vector Layers (Quelle: eigener Code)

```
//remove current coordinates, if existing
removeGeometryLayer();

// save the current Selection to global variable selectedOptions, so they can
only be changed with the apply button
var idField = document.getElementById("coordIDSelect").value;
var epsgField = document.getElementById("epsgInput").value;
selectedOptions.coordID = idField;
selectedOptions.epsg = epsgField;
```

Als nächstes wird überprüft, ob der Benutzer einen EPSG-Code für eine Koordinatentransformation eingegeben hat. Ist das nicht der Fall, werden standardmäßig geographische Koordinaten des WGS84 (EPSG: 4326) angenommen. Um alle standardisierten Koordinatensysteme zu unterstützen, wird mit dem EPSG-Code die Definition des dazugehörigen Koordinatensystem von epsg.io angefordert. Dies funktioniert über einen XMLHttpRequest an epsg.io, wobei die URL sich aus dem Schema + Hostnamen (https://epsg.io) und zusätzlich aus dem gewünschten EPSG-Code und dem Kürzel “.js“ für die proj4-Definition zusammensetzt (siehe Code 23). So wäre eine mögliche Anfrage:

```
https://epsg.io/32633.js
```

Die darauffolgende Rückmeldung von epsg.io:

```
proj4.defs("EPSG:32633","+proj=utm +zone=33 +datum=WGS84 +units=m +no_defs");
```

Wird keine Response erhalten, so wird dem Benutzer mitgeteilt, dass für den eingegebenen EPSG-Code keine Definition vorliegt. Wurde eine Response erhalten, so ist diese zwar ein fertiger Code zur Erstellung der Koordinatensystemdefinition, liegt jedoch nur als `string` vor. Dieser muss mit der Funktion `eval` ausgeführt werden (siehe Code 23).

Code 23 Anfrage einer Koordinatensystemdefinition von epsg.io über ein XMLHttpRequest (Quelle: eigener Code)

```
// If EPSG is not empty or 4326, the data has to be reprojected. get the .wkt
from epsg.io api
var responseString = "";
if (epsgField !== "4326" && epsgField !== "") {
  var xhrString = "https://epsg.io/" + epsgField + ".js";
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) { // when transformation is
    complete, insert the new coordinates
      var responseString = xhttp.response;
      if (responseString === "") {
        alert("The chosen coordinate system is unknown. Please check your
        input.");
      } else {
        // EPSG found
        eval(responseString); // execute the function, so projection
        is defined in proj4
      }
    }
  };
  xhttp.open("GET", xhrString, false); // not async
  xhttp.send();
}
```

Es wird nur die Definition des Koordinatensystems angefordert, die eigentliche Transformation geschieht clientseitig im Browser über die JavaScript-Bibliothek Proj4js. Auch eine komplette Transformation der Daten über ein XMLHttpRequest an epsg.io wäre möglich, allerdings ist die Länge der Query-URL mit der schnelleren “GET“-Funktion limitiert. Dies kann Probleme bei großen Daten mit mehreren tausend Messpunkten und bei der Transformation von Polygongeometrien hervorrufen, sowie die Performance der Applikation von der Qualität der Internetverbindung des Benutzers abhängig machen.

Wurde das Koordinatensystem erfolgreich definiert, oder liegen die Daten im System EPSG:4326 vor, welches von OpenLayers bereits standardmäßig unterstützt wird, kann mit der Darstellung der Punkte in der Karte begonnen werden. Der Geometrietyp

wird aus den Daten ausgelesen, das `styles`-Objekt enthält Stildefinitionen für Punkte und Polygongeometrien. So können beide Geometrietypen mit derselben Funktion bearbeitet werden.

Für das Lesen der Datensource wird eine `readFeatures`-Methode angewandt. In dieser werden die Daten mit der jeweiligen Koordinatensystemdefinition transformiert. OpenLayers erkennt, wenn ein Koordinatensystem in proj4 definiert wurde. Die Definition wird mit `ol.proj.get()` erhalten (siehe Code 24).

Code 24 Erstellen eines neuen VectorLayers in OpenLayers, Koordinatentransformation der Vector Source mit einer readFeatures-Methode (Quelle: eigener Code)

```
var geometryType = coords_json.features[0].geometry.type;
mapoch.currentFiles.geometryType = geometryType;
geometryLayer = new ol.layer.Vector({
  source: new ol.source.Vector({
    features: (new ol.format.GeoJSON({
      defaultDataProjection: 'EPSG:4326'
  })).readFeatures(coords_json, {
    dataProjection: ol.proj.get('EPSG:' + epsgField),
    featureProjection: 'EPSG:3857'
    })
  }),
  style: function(feature, resolution) {
    var geom = feature.getGeometry().getType();
    if (geom === 'Polygon') {
      geom = 'MultiPolygon'
    } //for easier styling, make polygons to multipolygons
    var id = feature.get(idField[1]);
    return styles[geom];
  }
});
var styles = { // initial style
  'Point': [new ol.style.Style({
    image: new ol.style.Circle({
      radius: 15,
      fill: new ol.style.Fill({
        color: 'black'
      })
    })
  })],
  'MultiPolygon': [new ol.style.Style({
    stroke: new ol.style.Stroke({
      color: 'white',
      width: 3
    }),
    fill: new ol.style.Fill({
      color: 'rgba(0, 0, 0, 0.8)'
      })
    })]
};
map.addLayer(geometryLayer);
```

Um den Benutzer ein möglichst gutes Feedback über den erfolgreichen Datenimport zu geben, wird auf die geladenen Daten gezoomt. Durch eine Animation wird dem Betrachter klarer, in welche Richtung sich die Kamera bewegt hat. Die Animation setzt sich zusammen aus Zoom (Veränderung des Zoomlevels) und Pan (Veränderung des Kartenmittelpunktes). Mit der Methode `map.getView().fit()` wird dafür gesorgt, dass der

Benutzer nach Abschluss der Animation sämtliche Features im Blick hat (siehe Code 24). Dadurch sollten mögliche Fehler beim Datenimport sofort ersichtlich sein (vgl. Abbildung 33).

Code 25 Animation aus Zoom und Pan zu den geladenen Daten (Quelle: eigener Code)

```
var zoom = ol.animation.zoom({
  resolution: map.getView().getResolution(),
  duration: 500,
  easing: ol.easing.inAndOut
});
// start the pan at the current center of the map
var pan = ol.animation.pan({
  source: map.getView().getCenter(),
  duration: 500,
  easing: ol.easing.inAndOut
});
map.beforeRender(zoom);
map.beforeRender(pan);
// when we set the center to the new location, the animated move will
// trigger the bounce and pan effects
var extent = geometryLayer.getSource().getExtent(); // zoom to all features
map.getView().fit(extent, map.getSize());
```

6.5 Einlesen, Verknüpfen und Darstellen der temporalen Messdaten

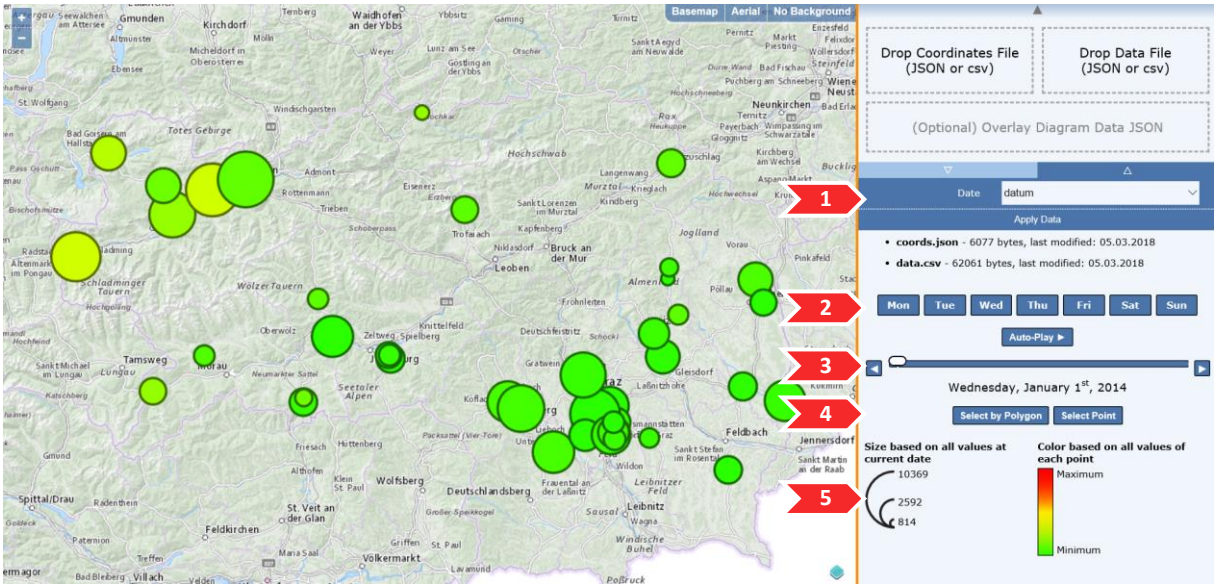


Abbildung 34 Einlesen der temporalen Daten und Verknüpfung mit den Messpunktfeatures. 1: Auswahl des Datum-Feldes, 2: Auswahl der Wochentage, 3: „Timeslider", 4: Selektion, 5: Legenden (Quelle: Eigene Abbildung)

In diesem Kapitel wird das Lesen und Verknüpfen der temporalen Daten mit den Messpunktkoordinaten behandelt. Das clientseitige Einlesen der csv oder JSON-Datei in den Browser erfolgt analog zu Kapitel 6.3, jedoch ist die einzige Eingabe, die vom Benutzer gefordert wird, die Auswahl der Zeitspalte. Als Ergebnis soll der Benutzer eine sinnvolle Darstellung der geotemporalen Daten samt Legende erhalten, sowie

Möglichkeiten zur Verfügung gestellt bekommen, diese Daten weiter zu inspizieren und zu vergleichen (siehe Abbildung 34).

Im JSON-Format wird ein `array` aus Objekten verlangt, wobei jedes dieser Objekte einen Zeitabschnitt darstellt. Die `properties` dieser Objekte bestehen zum einen aus einem key/value -Paar für die Zeit im Format ISO 8601, zum anderen aus jeweils einem key/value –Paar für jeden Messpunkt zu diesem Zeitpunkt, wobei der key den Namen des Messpunktes darstellt und der value ein Wert sein muss, der in eine JavaScript `number` umwandelbar ist (siehe Code 8, S. 62).

Das csv-Format wurde so definiert, dass jede Spalte einen Zeitpunkt darstellt, während jede Zeile den Messpunktnamen und die dazugehörigen temporalen Daten enthält (siehe Code 9, S. 63).

Analog dazu wäre eine feste Definition des Namens des „Zeit“-Properties im JSON-Format denkbar. Allerdings wurde dies, um das verlangte Datenformat möglichst kompatibel mit bereits bestehender Software zu halten, nicht umgesetzt. Der Benutzer sollte nicht zu viel Zeit in die Erstellung des verlangten Datenformates investieren müssen, sondern sollte, soweit möglich, mit bereits existenten Daten arbeiten können.

Da es sich beim JSON-Standard um ungeordnete Wertepaare handelt, ist auch eine definierte Lokalisierung des Zeit-Objektes (zum Beispiel als erstes `property` in jedem Objekt) nicht durchführbar. Mögliche Lösungsansätze sind das Benutzen von `arrays` oder ECMA 2015 `maps`[68].

6.5.1 TRANSFORMATION IN DATE OBJECTS

Nach Bestätigung des Datum-Feldes werden zuerst alle Zeitwerte, welche bis zu diesem Zeitpunkt als `string` vorliegen, in JavaScript `date objects` umgewandelt. Hierzu wird für jeden Zeitpunkt in den Daten der jeweilige String mit `string.substring()` zerschnitten und mit `new Date()` ein neues `date object` erstellt (siehe Code 26). Diese Objekte werden im globalen Objekt `mapoch` für die spätere Benutzung gespeichert.

Code 26 Erstellung der JavaScript date objects (Quelle: eigener Code)

```
for (i = 0; i < mapoch.zaehlstellen_data.length; i++) {
  var datestring = mapoch.zaehlstellen_data[i][selectedOptions.dateField];
  var thisYear = parseInt(datestring.substring(0, 4));
  var thisMonth = parseInt(datestring.substring(5, 7));
  var thisDay = parseInt(datestring.substring(8, 10));
  var thisDateComplete = new Date(thisYear, thisMonth - 1, thisDay); // JS-Date Month
  begins at 0
  mapoch.zaehlstellen_data[i][selectedOptions.dateField] = thisDateComplete;
}
```

[68] Mozilla and individual contributors (2018): Map https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

Zurzeit werden alle Zeiten im Format ISO 8601 akzeptiert (siehe Kapitel 6.3.1.2 *Dateiformate der Messwerte*), wobei nur bis auf die Ebene der Tage eingelesen wird. Genauere Unterteilungen wie Stunden, Minuten oder Sekunden sind nach demselben Schema möglich, falls die Datengrundlage eine solche Tiefe erfordert. In diesem Fall sollten auch der Timeslider und die Wochentagsauswahl angepasst werden, um die Daten für den Benutzer möglichst greifbar zu halten.

6.5.2 AUSWAHL DER WOCHENTAGE

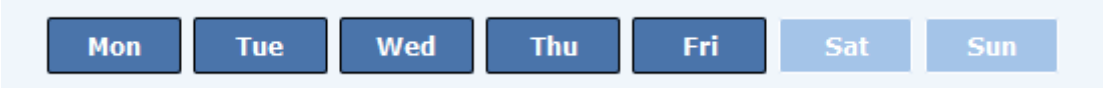


Abbildung 35 Buttons zur Auswahl der Wochentage. In diesem Fall sind die Tage Samstag und Sonntag inaktiv (Quelle: eigene Abbildung)

Durch das Erstellen von JavaScript `date objects` aus einfachen `strings` ist es möglich, weitere Informationen aus den Daten zu generieren, wie zum Beispiel den Wochentag. Durch eine Auswahl von Wochentagen ist es für den Benutzer einfacher, dieselben Tage oder gleiche Abschnitte der Woche zu vergleichen (Abbildung 35). So können bestimmte Werktage oder die Wochenenden von der Veränderung der Zeitkomponente ausgenommen werden.

In Mapoch wurde die Auswahl der Wochentage durch stark veränderte `HTML Input Elements` mit dem type-Attribut `checkbox` umgesetzt. Der `value` jedes Elements ist ein Integerwert von 0 bis 6, wobei 0 für Sonntag und 6 für Samstag steht. Diese Werte entsprechen den Rückgabewerten der Funktion `date.getDay()`, wodurch Abfragen möglich werden[69].

6.5.3 ERSTELLUNG DES TIMESLIDERS

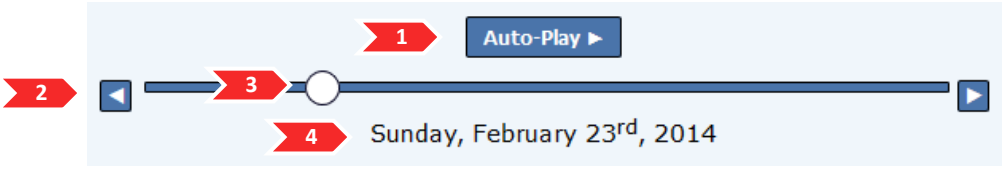


Abbildung 36 Timeslider zur Interaktion mit der Zeit-Komponente der Daten. 1: "Auto-Play"-Button (automatische Veränderung der Zeit im sekundentakt), 2: Button zur Veränderung um einen Zeitschritt, 3: Schieberegler, 4: Datumsanzeige (Quelle: eigene Abbildung)

Der „Timeslider“ ist ein Kontrollelement, das dem Benutzer die Steuerung durch die Zeit-Komponente der Daten ermöglicht. Der Benutzer kann entweder den Regler verschieben, mit den Buttons links und rechts des Reglers den Zeitpunkt um jeweils einen Schritt verändern, oder mit einem Auto-Play-Button den Zeitpunkt in einer Dauerschleife im Sekundentakt erhöhen (siehe Abbildung 36).

[69] MOZILLA AND INDIVIDUAL CONTRIBUTORS (2018): Date.prototype.getDay() https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getDay

Der Slider ist ein `HTML Input Element` mit dem `type`-Attribut `range`. Dieser wird bereits in der Datei `index.html` erstellt, wobei das Attribut `min` auf 0 gesetzt wird. Auch die Funktionen für das `onchange-` und `oninput-Event` wurden bereits definiert (siehe Code 27). Diese kommen zum Einsatz, wenn der Schieberegler entweder durch den Benutzer oder durch das Programm selbst (zum Beispiel während der auto-play Animation) aktiviert wird.

Code 27 Timeslider in index.html, min, onchange- und oninput-Funktionen sind bereits definiert. (Quelle: eigener Code)

```
<input type="range" id="time_slider" name="rangeInput" min="0"
onchange="updateInput(this.value, false);" oninput="updateInput(this.value, false);">
```

Nach dem Einlesen der Messdaten wird auch das Attribut `max` gesetzt. Die Attribute `min` und `max` definieren den Werteraum, den der `value` des Elements einnehmen kann. In diesem Fall wird max als `data.length-1` gesetzt, wobei `data.length` die Länge des `arrays` aller eingelesenen Messdaten ist (siehe Code 28). Somit stellt der `value` des Schiebereglers einen Integerwert für jeden Zeitpunkt in den Daten dar, beginnend mit 0. Wurden geotemporale Daten mit 100 Zeitpunkten eingelesen, kann das Element jeden Integerwert von 0 bis 99 annehmen.

Code 28 Setzen des Attributes "max" im Timeslider nach Einlesen der Daten (Quelle: eigener Code)

```
data = mapoch.zaehlstellen_data;
document.getElementById("time_slider").setAttribute("max", data.length -1);
```

Die drei Buttons zur Steuerung der Zeitkomponente lösen dieselbe Funktion `changeDateOneStop()` aus (siehe Code 29), wobei diese Funktion bei Aktivierung der Auto-Play-Buttons einmal jede Sekunde ausgelöst wird. Weiters wird im Auto-Play-Modus auch wieder von vorne begonnen, wenn der letzte Zeitpunkt erreicht wurde, während die anderen Buttons nur bis zum Ende der vorhandenen Daten benutzt werden können. Dadurch ist eine Dauerschleife, ähnlich einer Animation der Zeit (vgl. Kapitel 4.4.2: *Animierte Karten*), möglich.

Code 29 Funktion zum Verändern der Zeit um einen Zeitschritt. Ist step == -1, so wird auf der Zeitskala ein Schritt nach links, also zum nächstfrüheren Zeitpunkt, gesprungen. (Quelle: eigener Code)

```
function changeDateOneStep(step, loop)
  var x = document.getElementById("time_slider").value;
  var thisDate = parseInt(x) + parseInt(step); // thisDate = integer of Timestep (e.g. 0 =
  first Date in Data)
  var goLeft = (step == -1) ? true : false;
  updateInput(thisDate, goLeft, loop);
}
```

Durch jede Veränderung des Zeitpunktes, sei es durch den Timeslider oder durch einen der Buttons, muss der nächstmögliche Tag der aktivierten Wochentage (siehe

Abbildung 35) gefunden werden und der gesamte dargestellte Inhalt, sowohl in der Karte als auch in der Legende und in den erstellten Balken- und Kreisdiagrammen, upgedatet werden. Es wird solange danach gesucht, bis der nächste Zeitpunkt gefunden wird, der den Auswahlkriterien entspricht, oder kein einziger Wochentag ausgewählt wurde. Ist das Ende der geladenen Zeitreihe (beziehungsweise der Anfang, falls auf der Zeitskala nach links gesucht wird) erreicht, wird die Funktion abgebrochen. Diese Abbruchbedingung gilt jedoch nicht, falls die Schleife durch den Auto-Play-Button aktiviert wurde (siehe Code 30). Als Zeitpunkt wird in diesem Fall nicht der echte Zeitpunkt in den Daten gewählt, sondern der Integerwert des Timesliders.

Code 30 Suchen nach dem nächsten verfügbaren Tag, dessen Wochentag vom Benutzer gewählt wurde (slectedWeekdays) (Quelle: eigener Code)

```
while (foundNextWeekday == false) {
  thisDate = parseInt(thisDate); // parseInt because value of Timeslider is string
  if (thisDate > mapoch.zaehlstellen_data.length - 1) { // if maximum time is reached
    if (loop === true) {
      thisDate = 0;
    } else {
      break;
    }
  };
  if (thisDate < 0) {
    break;
  };

  var d = mapoch.zaehlstellen_data[thisDate].datum;
  if (typeof(selectedWeekdays) != "undefined" && selectedWeekdays.indexOf(d.getDay())>= 0)
  {
    updateStyle(thisDate);  //change appearance in map
    addPieCharts();  // change appearance of pie charts (if existing)

    foundNextWeekday = true;
    document.getElementById("time_slider").value = thisDate; // Update of Timeslider

    if (typeof selectedFeatures !== "undefined" && selectedFeatures.length > 0) {
      createPolyChart(selectedFeatures) //update column charts (if existing)
    }
  } else if (selectedWeekdays.length == 0) {
    alert("No Weekday Selected");
    break;
    foundNextWeekday = true;
  } // Break while when end of Data is reached
  else {
    thisDate = (goLeft == true) ? thisDate - 1 : thisDate + 1;
  }
}
```

6.5.4 MULTIDIMENSIONALE DARSTELLUNG DER GEOTEMPORALER DATEN

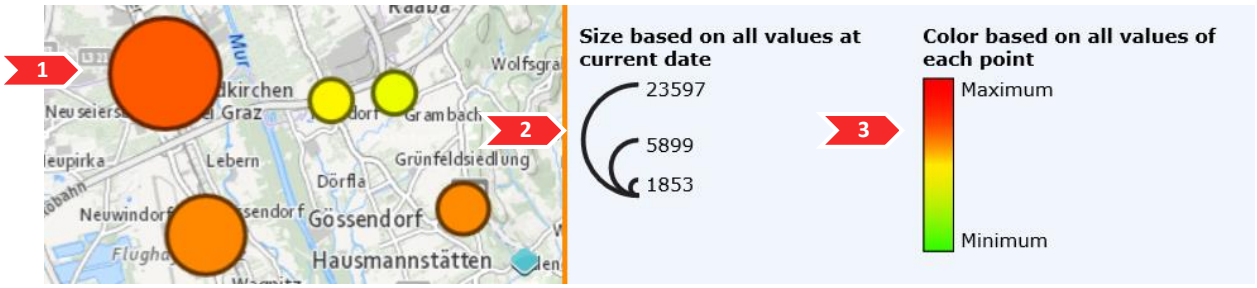


Abbildung 37 Darstellung der Messwerte über Größe und Farbton mit dazugehörige Legenden, 1: Kartenfeld mit Messpunkten, 2: Legende der Signaturengröße, 3: Legende der Signaturenfarbe (Quelle: eigene Abbildung)

Nach der Bestätigung des Datumfeldes durch den Benutzer und nach jeder Veränderung des Zeitpunktes werden die Daten neu dargestellt. Dies geschieht über zwei visuelle Variablen: die Größe und den Farbton (siehe Abbildung 37).

Der Farbton spiegelt den Messwert im Verhältnis zu allen Messwerten des jeweiligen Messpunktes zu allen Zeitpunkten wider. Je roter der Kreis, desto näher befindet sich der Messwert am Maximum aller gemessenen Werte an diesem Ort. Je grüner der Kreis, desto näher befindet sich der aktuelle Messwert am Minimum aller Messwerte am jeweiligen Ort.

In diesem Sinne bedeutet ein kleiner, roter Kreis, dass der Messwert zum gewählten Zeitpunkt zwar ein niedriger im Verhältnis zu den anderen Messwerten am selben Zeitpunkt ist, jedoch ein hoher Wert für den jeweiligen Messpunkt selbst. Durch das gleichzeitige Darstellen beider visuellen Variablen kann ein geübter Benutzer mit einem Blick Strukturen und Ausreißer in den Daten erkennen.

Die Größe des Kreises spiegelt das Verhältnis des Messwertes zu den Messwerten aller Messpunkte zum gewählten Zeitpunkt wider. Je größer der Kreis, desto höher ist der Wert an jenem Ort im Vergleich zu den Werten der anderen Messstellen zu dieser Zeit. Zur Berechnung der Größe der Kreise werden eben diese Werte aus den Messdaten abgefragt. Dies erfolgt über eine Schleife, aus der der größte gemessene Wert hervorgeht (siehe Code 31).

Code 31 Abfrage nach dem größten Messwert aller Messpunkte zu einem Zeitpunkt (Quelle: eigener Code)

```
var max_thisDay = -Infinity;
for (var k in mapoch.zaehlstellen_data[y]) { // for every point
  if (typeof mapoch.zaehlstellen_data[y][k] == 'number') { // only if number, not date
    var amount = mapoch.zaehlstellen_data[y][k];
    if (amount > max_thisDay) {
      max_thisDay = amount
      }; // maximum
    }
}
```

Damit die Karte lesbar bleibt, ist die Größe des größten Kreises konstant mit einem Radius von 35 Pixel festgelegt, unabhängig von den Eingangsdaten. Um die Flächeninhalte aller Kreise vergleichbar zu halten, wird aus dem größten Wert ein Skalierungsfaktor berechnet (siehe Code 32), mit dem die Darstellung der anderen Kreise angepasst wird.

Code 32 Berechnung des Skalierungsfaktors der Kreise (Quelle: eigener Code)

```
var maxAreaLargestCircle = Math.pow(35,2) * Math.PI; // Area of 35 Pixel Circle
var scaleFactor = maxAreaLargestCircle/max_thisDay;
```

Die Farbe der Features wird berechnet im Verhältnis zu den minimalen und maximalen Werten des jeweiligen Messpunktes zu allen Zeitpunkten. Diese Werte werden beim Einlesen einmal für jeden Messpunkt in einer Schleife berechnet und in einem `object` aus `arrays` gespeichert, wobei der erste Wert jedes `arrays` das Minimum und der zweite Wert das Maximum aller Messungen an diesem Punkt darstellt (siehe Code 33).

Code 33 Abfrage des Minimum- und des Maximumwertes eines Messpunktes (Quelle: eigener Code)

```
mapoch.min_max_zaehlstelle = {};
for (k = 0; k < Object.keys(data[0]).length; k++) { // name of zaehlstelle
  var name_zaehlstelle = Object.keys(mapoch.zaehlstellen_data[0])[k];
  if (name_zaehlstelle === dateField) {
    continue;
  }; //skip this if field is date field
  var min_max = [Infinity, -Infinity];
  for (i = 0; i < data.length; i++) {
    var amount = data[i][name_zaehlstelle];
    if (amount < min_max[0]) {
      min_max[0] = amount;
    };
    if (amount > min_max[1]) {
      min_max[1] = amount;
    };
}
mapoch.min_max_zaehlstelle[name_zaehlstelle] = min_max; // assign min/max-Values to Object
```

Die eigentliche „Verknüpfung“ der Geometrien der Karte mit den eingelesenen Messdaten erfolgt, indem das Aussehen der Punktgeometrien, in denen die Verknüpfungs-IDs gespeichert sind, mit den korrelierenden Daten in Form und Farbe verändert wird. Hierzu wird in eine OpenLayers `ol.vector.layer.setStyle()` –Funktion eine OpenLayers `ol.StyleFunction()` übergeben. Diese `StyleFunction` wird auf jedes Feature der Vektorlayers angewandt und kann so jedes Feature individuell graphisch verändern[70]. In der `StyleFunction` werden die Berechnungen der Stile für Punkte und Polygone definiert, allerdings wird die graphische Variable der Größe nur für Punktgeometrien verändert.

[70] OPENLAYERS (2018): ol.layer.vector https://openlayers.org/en/latest/apidoc/ol.layer.Vector.html

Innerhalb der `StyleFunction` wird der Geometrietyp abgefragt. Handelt es sich um Features des Typs „Polygon“, so werden sie wie „MultiPolygon“ behandelt, da deren Aussehen gleich sein soll. Anschließend wird der Name des Features abgerufen. Mit diesem ist es möglich, sowohl den aktuellen Messwert des Features zum gewählten Zeitpunkt (siehe Code 34), als auch die Minimum- und Maximum-Werte aller Messdaten dieses Messpunktes zu erhalten.

Code 34 StyleFunction 1: Abfrage der benötigten Werte (Quelle: eigener Code)

```
geometryLayer.setStyle(function(feature, resolution) {
  var geom = feature.getGeometry().getType(); // get geometry type
  if (geom === 'Polygon') {
    geom = 'MultiPolygon'
  } //for easier styling, make polygons to multipolygons

  var zaehlstelle = feature.get(mapoch.selectedOptions.coordID); //returns name of feature
  var amount = mapoch.zaehlstellen_data[y][zaehlstelle]; // returns value at current time

[…]
```

Da das menschliche Auge automatisch die Flächen zweier Objekte vergleicht, müssen die Messwerte die Flächeninhalte der Kreise darstellen. Daher wird jeder Messwert mit dem zuvor berechneten Faktor skaliert, um anschließend über die Kreisformel den benötigten Radius zu erhalten (siehe Code 35).

Code 35 StyleFunction 2: Berechnung des Radius jedes Kreises (Quelle: eigener Code)

```
[…]

  var scaledAmount = amount * scaleFactor;
  var radius_size = Math.sqrt(scaledAmount / (Math.PI));

[…]
```

Die Berechnung der Farbe geschieht nach dem HSL-Farbmodell. In diesem ist der Farbton („Hue“) als ein Farbkreis von 0-360 skaliert (siehe Abbildung 38). In diesem Farbkreis ist der bekannte Farbverlauf von Rot über Gelb nach Grün bereits vorhanden. Dadurch kann eine intuitive Farbkodierung (grün = gering, rot = hoch) ohne die Verwendung von „Look up Tables“ oder Drittsoftware einfach berechnet werden.

Abbildung 38 Darstellung von Hue im HSL-Farbmodell. In Mapoch werden nur die Werte von 0 bis 110 verwendet, wobei 0 (Rot) hoch und 110 (grün) niedrig bedeutet (Quelle: wikimedia.org)

Als geeignetster Farbraum werden die Werte von 0 bis 110 gewählt. Um zu garantieren, dass der Farbraum zur Gänze ausgenützt wird, muss der Messwert mit

allen Messwerten dieses Messpunktes zu allen Zeitpunkten skaliert werden (siehe Code 36).

$$Farbton = 110 - \left(110 * \frac{Messwert - Messwert_{min}}{Messwert_{max} - Messwert_{min}}\right)$$

Code 36 StyleFunction 3: Berechnung des Farbtons von Rot (0) bis Grün (110) (Quelle: eigener Code)

```
[…]

  var color_hue = 110 - Math.round(((amount-min_max_zaehlstelle[zaehlstelle][0]) / (min_max_zaehlstelle[zaehlstelle][1]-min_max_zaehlstelle[zaehlstelle][0])) * 110)

[…]
```

Sind Radius und Farbwert berechnet, so kann das `ol.style` Objekt erstellt werden, welches von der `StyleFunction` zurückgegeben wird. Für Punktgeometrien werden Kreise mit dem berechneten Radius und der berechneten Farbe erstellt, die Umrandung jedes Kreises wird als dunklere Farbe desselben Farbtons definiert. Die Polygone beziehungsweise MultiPolygone werden lediglich mit der berechneten Farbe befüllt, die Größe bleibt unskaliert (siehe Abbildung 39). Zurückgegeben wird der `ol.style` der Geometrie des Objektes (siehe Code 37).

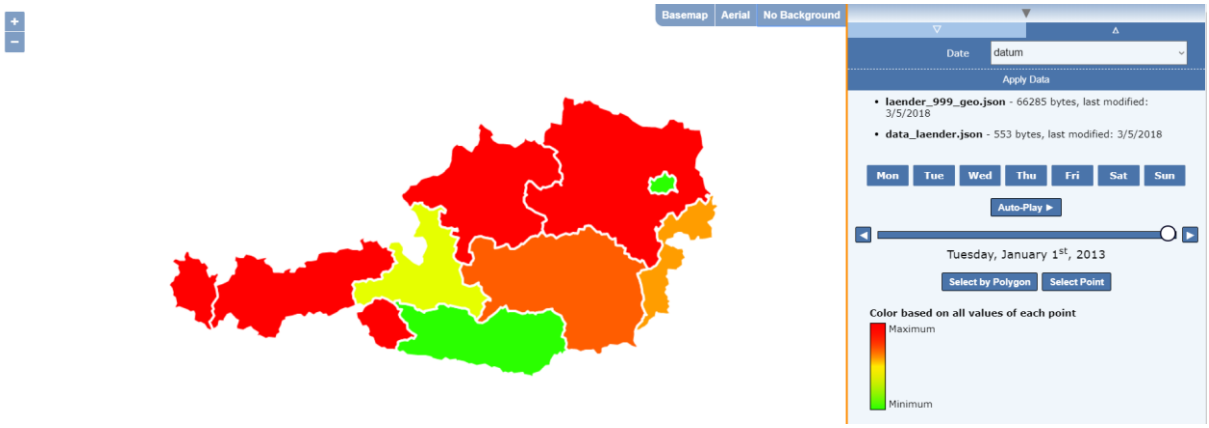


Abbildung 39 Darstellung von geotemporalen Daten von Polygongeometrien. Der Fabton ist die einzige verwendete graphische Variable (Quelle: eigene Abbildung)

Code 37 StyleFunction 4: Erstellung der ol.style-Objekte mit den berechneten Werten (Quelle: eigener Code)

```
[…]

  var styles = {
    'Point': [new ol.style.Style({
      image: new ol.style.Circle({
        radius: radius_size,
        fill: new ol.style.Fill({
          color: 'hsl(' + color_hue + ', 100%, 50%)'
        }),
        stroke: new ol.style.Stroke({
          color: 'hsl(' + color_hue + ', 100%, 20%)',
          width: 3
        })
      })
    })],
    'MultiPolygon': [new ol.style.Style({
      stroke: new ol.style.Stroke({
        color: 'white',
        width: 3
    }),
    fill: new ol.style.Fill({
      color: 'hsl(' + color_hue + ', 100%, 50%)'
    })
    })]
  };
  return styles[geom];
});
```

6.5.5 ERSTELLUNG DER LEGENDE (PUNKTGEOMETRIEN)

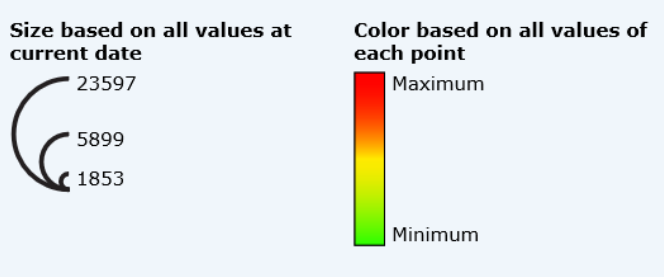


Abbildung 40 Legende für Größe und Farbe der Punktgeometrien (Quelle: eigene Abbildung)

Die Legenden der Punktfeatures sind zwei statische Bilder, da sich die maximale Größe der Kreise und die Farbskala nicht verändern. Dynamisch ist allerdings die Beschriftung der Größenlegende, da diese Werte variabel sind und sich mit jedem Zeitpunkt der Daten ändern können. Dies bedeutet jedoch auch, dass zwei gleich große Kreise zu zwei verschiedenen Zeitpunkten nicht unbedingt denselben Messwert aufweisen müssen. Zum intuitiveren Vergleich von Messdaten zu verschiedenen Zeitpunkten wird die Möglichkeit bereitgestellt, die Messpunkte zu selektieren, die Daten als Balkendiagramm anzuzeigen und temporär zu speichern. Näheres dazu im Kapitel 6.6: *Selektion von Features und Generierung von Balkendiagrammen*.

Der größte Kreis in Abbildung 40 entspricht dem größten Wert aller Messpunkte zum aktuellen Zeitpunkt und ist somit immer in der Karte als Objekt vorhanden. Die anderen Kreisgrößen der Legende sind nicht zwingend in der Karte vorhanden, sie dienen dem Benutzer ausschließlich dazu, um einen groben Eindruck über die Größenverhältnisse zu vermitteln. Da die Kreisflächen zueinander quadratisch abhängig sind, ist die Berechnung der Flächeninhalte durch einfache Division möglich (siehe Code 38).

Code 38 dynamische Beschriftung der Größentabelle (Quelle: eigener Code)

```
if (mapoch.currentFiles.geometryType !== 'Polygon') {
  // biggest circle (d=70px) = maximum value
  document.getElementById("size_image_max").innerHTML = max_thisDay;
  var middle_value = Math.round(max_thisDay / 4); // Circle with half diameter
  document.getElementById("size_image_mid").innerHTML = middle_value;
  var small_value = Math.round(max_thisDay * 0.07854); // Circle with 1/7 diameter
  document.getElementById("size_image_min").innerHTML = small_value;
}
```

6.5.6 ERSTELLUNG OPTIONALER PIE CHARTS ALS POLYGON-OVERLAY

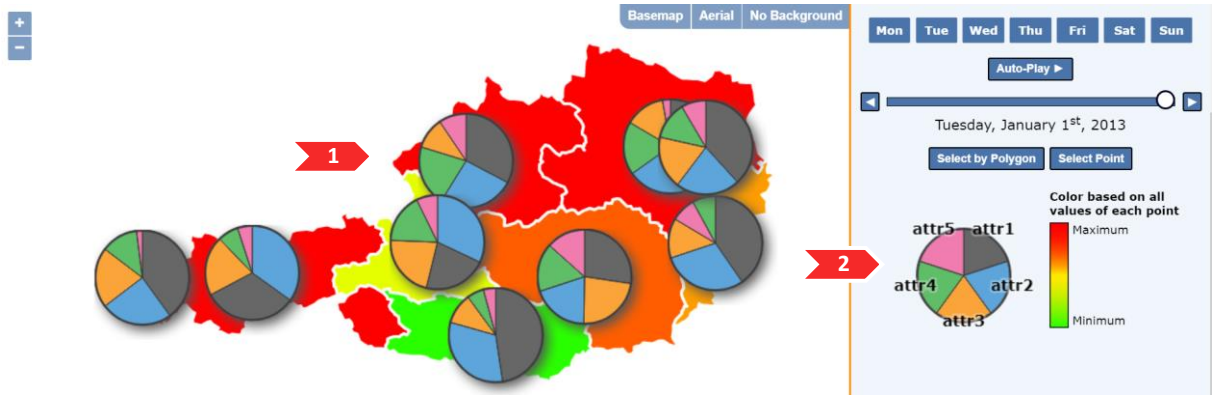


Abbildung 41 Pie Charts als Polygon-Overlays, 1: Kartenfeld mit Polygondaten und Pie Chart Daten, 2: Legende der Farben der Pie Chart Attribute und der Polygone (Quelle: eigene Abbildung)

Optionale Pie Charts als Overlay von Polygondaten können dem Benutzer in einer Vielzahl an Anwendungen Mehrwert generieren. Ein klassisches Beispiel sind Wahlergebnisse, die pro Wahlsprengel als prozentuelle oder absolute Ergebnisse für jede Partei vorliegen. Die Farbe der Polygone könnte im Hintergrund die prozentuelle Wahlbeteiligung visualisieren. Diese Art der Darstellung ermöglicht keine genaue Aussage über den Wahlausgang einer Wahl als Ganzes, kann jedoch die Wahlergebnisse und Wahlbeteiligung mehrerer Wahlen untereinander vergleichbar und geographische Muster und Strukturen erkennbar machen.

Weitere Anwendungsgebiete für diese Funktion können in den Bereichen Tourismus, Landwirtschaft, Sozialgeographie und Wirtschaftsgeographie gefunden werden.

Das Datenformat für die Zusatzfunktion der Pie Charts Overlays ist enger bestimmt als jenes der Grundfunktionen von Mapoch. Erlaubt sind nur Daten im JSON-Format,

wobei der `key` des Zeitfeldes mit dem `key` der Messdaten der Polygone übereinstimmen muss. Für jeden Zeitpunkt muss ein Objekt mit dem Namen jedes geographischen Features vorhanden sein. Die `properties` dieser Objekte stellen die Datengrundlage für die Pie Charts dar. Jedes Objekt muss dafür dieselben `properties` aufweisen können.

Code 39 Beispiel-JSON für Pie Chart Overlays (Quelle: eigener Code)

```
[{
  "datum":"2011-01-01",
  "Oberösterreich":{"attr1": 144, "attr2": 80, "attr3": 53, "attr4": 44, "attr5": 19 },
  "Tirol":{"attr1": 100, "attr2": 77, "attr3": 57, "attr4": 60, "attr5": 14 },
  "Steiermark":{"attr1": 111, "attr2": 64, "attr3": 51, "attr4": 77, "attr5": 10 }
},
{
  "datum":"2012-01-01",
  "Oberösterreich":{"attr1": 140, "attr2": 70, "attr3": 33, "attr4": 44, "attr5": 9 },
  "Tirol":{"attr1": 80, "attr2": 77, "attr3": 57, "attr4": 50, "attr5": 14 },
  "Steiermark":{"attr1": 71, "attr2": 64, "attr3": 51, "attr4": 77, "attr5": 30 }
}]
```

Das Einlesen erfolgt über Drag&Drop analog zur den in Kapitel 6.3 und 6.4 beschriebenen Vorgehensweisen, wobei durch das genau vorgegebene Datenformat keine weitere Benutzereingabe erforderlich ist.

Der Größenvergleich von Kreissegmenten ist für den Menschen äußerst schwierig, da das Problem unbewusst auf einen Vergleich von Winkeln reduziert wird[71]. Um den Betrachter eine Hilfestellung zu bieten, hat sich in der Diagrammdarstellung die Konvention durchgesetzt, am nördlichsten Punkt des Kreises mit dem größten Kreissegment zu beginnen und im Uhrzeigersinn der Größe nach die restlichen Kreisdiagramme anzuordnen. Da nicht davon ausgegangen werden kann, dass die eingelesene Datengrundlage der Pie Chart-Elemente in ihrer Summe ein logisches Ganzes ergeben, wären Balkendiagramme in vielen Fällen angemessener. Diese können jedoch eine Vielzahl kartographischer Probleme hervorrufen.

Da also die Merkmale jedes Pie Charts der Größe nach angeordnet werden, müssen sich gegebenenfalls die Farbreihenfolgen der Kreissektoren gemäß der Datengrundlage anpassen. Um dies zu gewährleisten, wurde beim Einlesen der Daten eine globale „Color Map“ angelegt, welche die Merkmale eindeutig einer Farbe zuweist. Als Farbschema wird FEW'S Farbpalette „medium“ verwendet (siehe Tabelle 6, S. 34 beziehungsweise FEW 2012, S. 344), wobei der graue Farbton unter Berücksichtigung der verwendeten Basemaps leicht verändert wurde.

[71] KOSARA, R. (2010): Understanding Pie Charts. https://eagereyes.org/techniques/Pie Charts

Code 40 Erstellung einer Color Map (Quelle: eigener Code)

```
var PieChartColors = ['#676767', '#5DA5DA', '#FAA43A', '#60BD68', '#F17CB0',
'#B2912F','#B276B2','#DECF3F','#F15854']; // Colors of each slice of the Pie Chart
var i = 0;
for (var key in PieChartDataElement) {
  mapoch.PieChartColorMap[key] = PieChartColors[i++];
}
```

Die Pie Charts werden ohne Verwendung einer Bibliothek als HTML5 Canvas Element erzeugt. Diese Elemente können als DOM-Objekte erstellt werden und später in die OpenLayers-Karte als Overlays zu bestimmten Punkten eingefügt werden. Bevor mit der eigentlichen Erstellung des Kreisdiagrammes begonnen werden kann, muss die Summe aller Werte der Diagrammdaten erstellt werden (siehe Code 41). Dies ist notwendig, um die absoluten Werte der gegebenen Daten als prozentuelle Anteile des Kreises darzustellen. Sollten sich in den Eingangsdaten bereits Prozentwerte befinden, werden sie durch diese Transformation nicht verändert.

Code 41 Summierung aller Werte der Pie Chart Daten eines Punktes (Quelle: eigene Abbildung)

```
var myTotal = 0;
for (var key in dataObject) {
  myTotal += dataObject[key];
};
```

Anschließend wird ein quadratisches `canvas` Element mit einer Größe von 115 Pixel erstellt. Das Zentrum des Kreisdiagrammes befindet sich bei der Koordinate 50/50, wobei sich der Ursprung des Koordinatensystems in der linken oberen Ecke befindet. Diese Platzierung lässt ausreichend Platz für die Erstellung eines Schattens rechts unten (siehe Abbildung 42). Der Schatten erzeugt die Illusion eines Höhenunterschiedes zum Hintergrund, von dem sich die Kreisdiagramme wortwörtlich abheben sollen. Hierzu wird ein Bogen (`arc`) mit einem Schatten gezogen, der Mittelpunkt ist der Mittelpunkt des Kreisdiagrammes, der Radius ist der Radius des Kreisdiagrammes, der Startwinkel ist 0 und der Endwinkel ist 2 Pi (360° in Radiant) (siehe Code 42).

Abbildung 42 Schatten eines Pie Charts für einen optischen Abstand zum Hintergrund (Quelle: eigene Abbildung)

Code 42 Erstellung eines quadratischen Canvas und des Schlagschattens eines Kreises (Quelle: eigener Code)

```
var canvas = document.createElement('canvas');
canvas.setAttribute('width','115');
canvas.setAttribute('height','115');
var ctx = canvas.getContext("2d");
var lastend = - Math.PI / 2; //quarter circle in radians so piechart starts north

// hardcoded center, canvas is larger because of shadow
var centerX = 50;
var centerY = 50;

//draw a dropshadow so the charts are "floating" above the map
ctx.save(); // save style of context, so that drop shadow is not applied on every single
element
ctx.shadowColor = "#555555";
ctx.shadowBlur = 20;
ctx.shadowOffsetX = 5;
ctx.shadowOffsetY = 5;
ctx.fillStyle = "white";
ctx.beginPath();
ctx.arc(centerX, centerY, centerY,0,2*Math.PI);
ctx.closePath();
ctx.fill();
// restore the saved canvas state
ctx.restore();
```

Nun werden die Werte der Merkmale der Größe nach geordnet. Jeder dieser Werte wird mit 2 Pi multipliziert und durch die vorher bestimmte Summe der Werte dividiert, um den Zentriwinkel des Kreissektors in Radiant zu erhalten. Dieser berechnete Winkel ergibt, zusammen mit dem Endwinkel des letzten Segments, die Form des aktuellen Kreissegments. (siehe Code 43 und Abbildung 43)

Code 43 Erstellung der Kreissegmente der Pie Charts (Quelle: eigener Code)

```
// get the key sorted by their values for the current object
var sortedValuesKeys = sortValues(dataObject);

for (var i = 0; i < sortedValuesKeys.length; i++) {
  ctx.fillStyle = mapoch.PieChartColorMap[sortedValuesKeys[i]]; // look up the color f
  or the current key
  ctx.beginPath();
  ctx.moveTo(centerX, centerY);
  // Arc Parameters: x, y, radius, startingAngle (radians), endingAngle
  // (radians), antiClockwise (boolean)
  ctx.arc(centerX, centerY, centerY, lastend, lastend + (Math.PI * 2 *
  (dataObject[sortedValuesKeys[i]] / myTotal)), false);
  ctx.lineTo(centerX, centerY);
  ctx.fill();
[…]
```

Um den Farbkontrast zwischen den einzelnen Elementen aufrecht zu erhalten, wird eine dünne Linie zwischen jedem Kreissegment gezogen. Danach wird der Endwinkel des gezeichneten Segmentes gespeichert, welches für das nächste Segment den Anfangswinkel darstellt (siehe Code 44).

Abbildung 43 Erstellung eines Kreissegmentes mit Trennlinie zum nächsten Segment (Quelle: eigene Abbildung)

Code 44 Erstellung der Kontrastlinie zwischen den Kreissgmenten (Quelle: eigener Code)

```
[…]
  //make a thin line between every segment
  ctx.moveTo(centerX, centerY);
  ctx.lineWidth="1";
  ctx.strokeStyle="#3c3c3c";
  ctx.lineTo(centerX + 50 * Math.cos(lastend), centerY + 50 * Math.sin(lastend));
  ctx.stroke();
  lastend += Math.PI * 2 * (dataObject[sortedValuesKeys[i]] / myTotal);
};
```

Nach Abschluss aller Segmente wird eine etwas dickerer Kreis um das gesamte Diagramm gezeichnet. Diese soll einen höheren Kontrast zum Hintergrund erzeugen und dadurch vor möglichen graphischen Störeinflüssen schützen (siehe Code 45 und Abbildung 44).

Abbildung 44 dickerer Kreis um alle Segmente um den Kontrast zum Hintergrund zu erhöhen und die Ecken der Kreissegmente zu glätten (Quelle: eigene Abbildung)

Code 45 Border um das gesamte Diagramm, um den Einfluss der Basemap zu minimieren (Quelle: eigene Abbildung)

```
//when done, create a 360’ arc as a border
ctx.beginPath();
ctx.lineWidth="2";
ctx.strokeStyle="#3c3c3c";
ctx.arc(centerX, centerY, centerY-1,0,2*Math.PI);
ctx.stroke();

return(canvas);
```

Beim Einlesen der Diagrammdaten wird der gesamte Vorgang einmalig für die Erstellung einer Legende ausgeführt. In der Legende weisen alle Spalten dieselbe Größe auf, zusätzlich sind sie mit den ausgelesenen Namen der Merkmale ausgestattet.

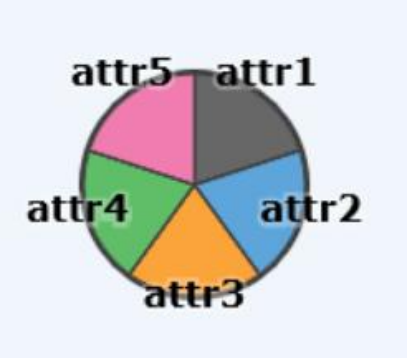


Abbildung 45 Dynamisch erstellte Legende der Kreisdiagramme mit Merkmalsnamen (Quelle: eigene Abbildung)

Um die Pie Charts der Karte hinzuzufügen, müssen sie geocodiert werden. Da Mapoch dieses Feature nur für Polygongeometrien anbietet, wird der Ankerpunkt jedes Diagramms als Zentrum des zugehörigen Polygons bestimmt. Handelt es sich beim Geometrietyp um `MultiPolygon`, so wird das Zentrum des flächenmäßig größten Teilpolygons stattdessen verwendet. Meistens ist das größte Polygon jene Fläche mit dem größten Repräsentationswert des Features. Ein Beispiel dafür ist Tirol mit den Exklaven Osttirol und Jungholz.

Die technische Umsetzung erfolgt über OpenLayers-Funktionen. Das Extrahieren der vorhandenen Polygone der Karte erfolgt über `ol.layer.vector.getSource().getFeatures()`, das Extrahieren aller Einzelpolygone der `MultiPolygon`-Features über `ol.feature.getGeometry().getPolygons()`, die Flächenbestimmung über `ol.geom.Polygon.getArea()` und die Generierung des Ankerpunktes über `ol.geom.Polygon.getInteriorPoint()` (siehe Code 46). Diese Funktion generiert einen Punkt, welcher mit Gewissheit im Polygon liegt, in den meisten Fällen ist dies der Schwerpunkt.

Code 46 Generierung der Ankerpunkte der Pie Charts (Quelle: eigene Abbildung)

```
var allPolygons = matchedFeature.getGeometry().getPolygons();
var largestPolygon;
var largestPolygonArea = 0;
allPolygons.forEach(function(thisPolygon){
  var thisArea = thisPolygon.getArea();
  if (thisArea > largestPolygonArea) {
    largestPolygon = thisPolygon;
    largestPolygonArea = thisArea;
  };
centerPoint = largestPolygon.getInteriorPoint();
```

Nachdem der Ankerpunkt gefunden wurde, kann ein neues `ol.Feature` erstellt werden. Als Geometrie wird der eben berechnete Zentrumspunkt angegeben, als `Style` das zugehörige Kreisdiagramm, wobei der Offset in X- und Y-Richtung jeweils dem Radius des Kreises entspricht (siehe Code 47).

Code 47 Erstellung eines ol.Feature mit Aussehen des erstellten Pie Charts (Quelle: eigener Code)

```
var iconFeature = new ol.Feature({
  geometry: centerPoint,
  name: 'pieChart'+key
});
//create canvas element as style
var iconStyle = new ol.style.Style({
  image: new ol.style.Icon(({
    anchor: [50, 50],
    anchorXUnits: 'pixels',
    anchorYUnits: 'pixels',
    img: mapoch.PieChartCanvasElements[key],
    imgSize: [115, 115]
  }))
});
```

Nachdem eine `ol.Source` mit den Ankerpunkten und Styles befüllt wurde, werden die Pie Charts als neuer Vektorlayer der Karte hinzugefügt. Die gesamte Routine wird bei jedem Wechsel der Zeitkomponente durchgeführt, da auch die Pie Chart-Daten eine temporale Ausprägung besitzen und sich mit der Karte verändern müssen.

6.6 Selektion von Features und Generierung von Balkendiagrammen

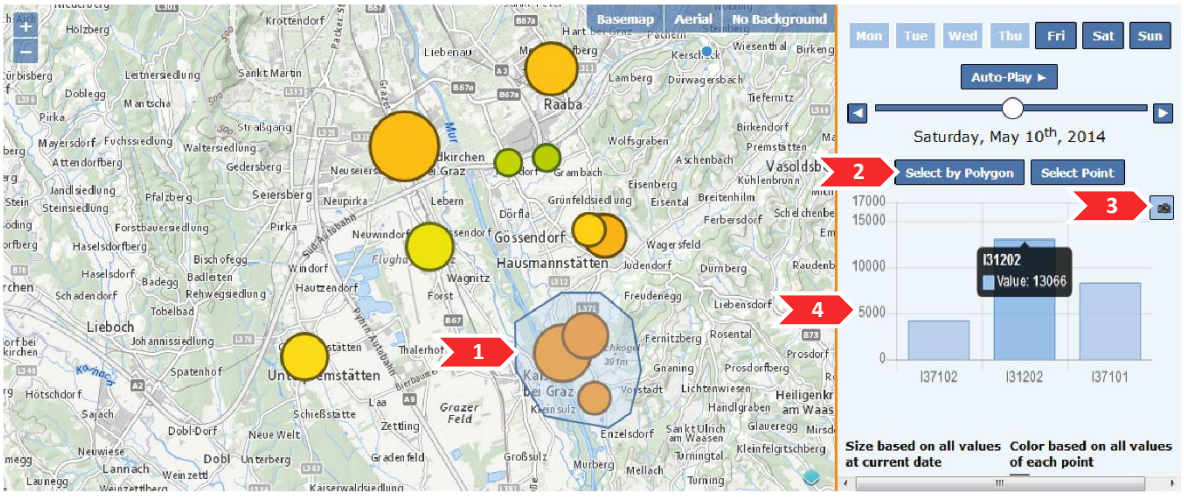


Abbildung 46 Selektion von Messpunkten. 1: Polygonselektion in der Karte, 2: Selektion über Polygon oder Einzelselektion, 3: Speichern von Selektion und Zeitpunkt, 4: Messdaten der selektierten Elemente als Balkendiagramme (Quelle: eigene Abbildung)

Eine benutzergesteuerte Selektion von Features in der Karte ist zwingend notwendig, um die genauen Messwerte der eingelesenen Daten zu kommunizieren. In diesem Sinne werden klassische Balkendiagramme der Messwerte der selektierten Features generiert, um eine absolute Vergleichbarkeit der Werte zu gewährleisten und somit den Problemen des Größenvergleiches von Kreisgeometrien entgegenzuwirken. Zusätzlich soll es dem Benutzer ermöglicht werden, Selektionsstatus (selektierte

Elemente und Zeitpunkt) als „Snapshots“ zwischenzuspeichern, um Messwerte verschiedener Zeitpunkte direkt zu vergleichen.

Die Selektion in der Karte wird mit der OpenLayers-Interaktion `ol.Select` umgesetzt. Es werden sowohl Selektion über gezeichnete Polygone als auch Punkt-Selektion umgesetzt. Im Folgenden wird die Polygon-Selektion beschrieben, die Punkt-Selektion erfolgt analog dazu.

Durch das Betätigen des Buttons zu Polygonselektion werden etwaige alte Selektionen, Selektions-Interaktionen und Zeichenlayer gelöscht. Anschließend wird ein neuer Vektorlayer als Zeichenlayer erstellt und die neue Interaktion `ol.interaction.Draw` des Typs `Polygon` wird initialisiert (siehe Code 48).

Code 48 Polygonselektion 1: Entfernen der alten Interaktion und Initialisierung einer neuen ol.interaction.Draw (Quelle: eigener Code)

```
Mapoch.draw;
function SelectByPolygon() {
  // remove point selection
  if (typeof(mapoch.select) !== "undefined") {
    mapoch.select.getFeatures().item(0).setStyle(null)
    map.removeInteraction(mapoch.select);
  };
  if (typeof(mapoch.draw) !== "undefined") {
    map.removeInteraction(mapoch.draw);
    mapoch.drawingSource.clear();
  };
  if (typeof(mapoch.drawingSource) !== "undefined") {
    mapoch.drawingSource.clear();
  }
  mapoch.drawingSource = new ol.source.Vector();

  var drawingLayer = new ol.layer.Vector({
    source: mapoch.drawingSource,
    style: new ol.style.Style({
      fill: new ol.style.Fill({
        color: 'rgba(191, 214, 239, 0.4)'
      }),
      stroke: new ol.style.Stroke({
        color: '#4A74AA',
        width: 2
      })
    })
  });
  map.addLayer(drawingLayer);

  mapoch.draw = new ol.interaction.Draw({
    source: mapoch.drawingSource,
    type: 'Polygon'
  });
[…]
```

Nach der Initialisierung der `Draw` Interaktion können Listener Funktionen der Events `drawstart` und `drawend` definiert werden. Bei einem `drawstart`-Event, also bei Zeichenbeginn jedes neuen Polygons, wird die aktuelle Selektion gelöscht. Bei einem `drawend`-Event, also nach Abschluss eines Polygons, wird nach allen Geometrien in der

Karte gesucht, welche das vom Benutzer gezeichnete Polygon schneiden (siehe Code 49). Über diese selektierten Features werden gemeinsam mit dem `value` des Timesliders die Balkendiagramme erstellt.

Überprüft wird die Schnittbedingung über den `Extent` der Features, welche bei Punktgeometrien der Punkt selbst darstellt und bei Polygongeometrien der Envelope. Dadurch können mit derselben Funktion beide Geometrietypen abgedeckt werden.

Code 49 Polygonselektion 2: Definition von drawstart- und drawend-Funktionen (Quelle: eigener Code)

```
[…]
  mapoch.draw.on('drawstart', function(e) {
    mapoch.drawingSource.clear();
  });

  draw.on('drawend', function(e) {
    var polygonGeometry = e.feature.getGeometry();
    mapoch.selectedFeatures = []; // Array for Features
    mapoch.oldSelectedStreetNames = [] // Array for street names, if same amount of
points are selected, but different streetnames -> redraw chart completely

    // for every feature...
    for (i = 0; i < geometryLayer.getSource().getFeatures().length; i++) {
      var pointExtent =
      geometryLayer.getSource().getFeatures()[i].getGeometry().getExtent();

      //returns true when Polygon intersects with Extent of Point (= Point itself)
      if (polygonGeometry.intersectsExtent(pointExtent) == true) {
        mapoch.selectedFeatures.push(geometryLayer.getSource().getFeatures()[i]);
      }
    }
    createPolyChart(mapoch.selectedFeatures);
  });
  map.addInteraction(mapoch.draw);
}
```

Die Balkendiagramme werden mit der Software chart.js erstellt (siehe Kapitel 2.2.1.1: *Chart.js*). Wenn ein Balkendiagramm erstellt werden soll, jedoch dieselben Features selektiert sind, soll dieses nur upgedatet werden. Ansonsten soll es gelöscht und komplett neu geladen werden. Das bloße Updating der Balkendiagramme ermöglicht Animationen der einzelnen Balken. Dadurch sieht der Benutzer sofort, ob ein Wert nach dem Zeitsprung gestiegen oder gefallen ist.

Um zu erkennen, ob ein simples Updating der Werte im Diagramm möglich ist, werden die Namen der selektierten Objekte extrahiert und mit den gespeichterten Namen der letzten Selektion verglichen. Sind sie nicht ident wird das alte Diagramm gelöscht (siehe Code 50).

Code 50 Überprüfung, ob ein einfaches Updating der Werte im Balkendiagramm möglich ist (Quelle: eigener Code)

```
var selectedStreetNames = [];
for (i = 0; i < selectedFeatures.length; i++) {
  selectedStreetNames.push(selectedFeatures[i].getProperties()[
  mapoch.selectedOptions.coordID]); // get all Measurement Points from selection
};

var SameStreetNames = selectedStreetNames.equals(mapoch.oldSelectedStreetNames);
var destroyChart = false;
if (myChart.id !== "myChart" && !SameStreetNames) {
    myChart.destroy();
    destroyChart = true;
}
oldSelectedStreetNames = selectedStreetNames;
```

Ist ein Updating möglich, so muss nur die Datengrundlage der bereits vorhandenen Balken verändert und die Grafik mit Chart.js-Funktionen upgedatet werden (siehe Code 51). Wenn nicht dediziert deaktivert, werden Animationen und andere Effekte automatisch aktiviert.

Code 51 Updating eines vorhandenen Chart.js Balkendiagramm (Quelle: eigener Code)

```
if (myChart.id !== "myChart" && destroyChart == false && selectedFeatures.length !== 0) {
  console.log("chart already exists, redraw with new values");
  myChart.labels = selectedStreetNames;
  myChart.data.datasets[0].data = selectedData;
  myChart.update();
  myChart.render();
  myChart.resize();
[…]
```

Ist dieses Updating nicht möglich, entweder, weil noch kein Diagramm existiert oder weil die selektierten Messpunkte geändert wurden, muss das Diagramm neu erstellt werden. Als Maximum der Y-Achse wird der oben berechnete Maximalwert der selektierten Messpunkte zu allen Zeitpunkten herangezogen, die Labels entsprechen den Namen der selektierten Messstellen (siehe Code 52).

Code 52 Neuerstellung des Chart.js-Balkendiagramms (Quelle: eigener Code)

```
} else if (selectedFeatures.length !== 0) { // If Chart didn‘t exist before...
  var ctx = document.getElementById("myChart");
  myChart = new Chart(ctx, {
  type: 'bar',
  data: {
    labels: selectedStreetNames,
    datasets: [{
      label: 'Value',
      data: selectedData,
      backgroundColor: 'rgba(164, 196, 232, 0.7)',
      borderColor: 'rgba(    74, 116, 170, 0.7)',
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      yAxes: [{
        ticks: {
          min: 0,
          max: dataMax,
          beginAtZero: true
        }
      }]
    },
    legend: {
      display: false
    }
  }
});
```

Durch einen in der `index.html` erstellten Button im rechten oberen Eck des Balkendiagrammes wird eine Funktion zur Erstellung der Snapshot-Buttons aufgerufen. Diese nimmt die aktuelle Zeitkomponente in der Form des `value` des Timesliders und die aktuell selektierten Features in ein Array `snapshotArray`. Im Button selbst wird die Anzahl der Snapshots bei der Erstellung des aktuellen Buttons gespeichert. Ist der erstellte Snapshot zum Beispiel der dritte erstellte Snapshot, so befinden sich die Informationen über die Zeit und die selektierten Features an dritter Stelle des `snapshotArray` (siehe Code 53).

Code 53 Speichern der Zeit und der selektierten Features in ein Array in Erstellung eines Buttons, um diese Ansicht wiederherzustellen (Quelle: eigener Code)

```
function snapshot() {
    if (typeof(mapoch.snapshotArray) == "undefined") {
        mapoch.snapshotArray = [];
    };

    // create array with parameters of this snapshot
    var thisSnapshot = [];
    // Save Current date
    thisSnapshot[0] = parseInt(document.getElementById("time_slider").value);
    // Save Current Selected Features
    thisSnapshot[1] = selectedFeatures;
    mapoch.snapshotArray.push(thisSnapshot);

    // append row to the HTML table
    var tbl = document.getElementById('snapshot_table') // table reference
    var row = tbl.insertRow(tbl.rows.length) // append table row
    var eyeButtonCell = row.insertCell(0);

    var buttonText = "Snapshot " + tbl.rows.length;
    // create button with value of index of array (of this snapshot)
    var btn = document.createElement('input');
    btn.type = "button";
    btn.className = "other_button";
    btn.setAttribute("id", "showSnapshot");
    btn.value = buttonText;
    btn.setAttribute('snapshotIndex', tbl.rows.length);
    btn.onclick = function() {
        showSnapshot(this.getAttribute('snapshotIndex') - 1);
    };

    eyeButtonCell.appendChild(btn);

    document.getElementById("snapshot_div").style.visibility = 'visible';

};
```

Mit diesen Informationen können Ansichten der Webapplikation Mapoch kurzzeitig gespeichert und jederzeit wieder abgerufen werden, bis neue Daten geladen werden. Diese Funktion ist besonders beim direkten Vergleich von zwei nicht-aufeinanderfolgenden Zeitpunkten, wie etwa denselben Tag in zwei verschiedenen Jahren. Durch die Aktivierung des „Delete Snapshots“-Buttons können alle erstellten Snapshots gelöscht werden, um Platz für neue zu schaffen (siehe Abbildung 47).

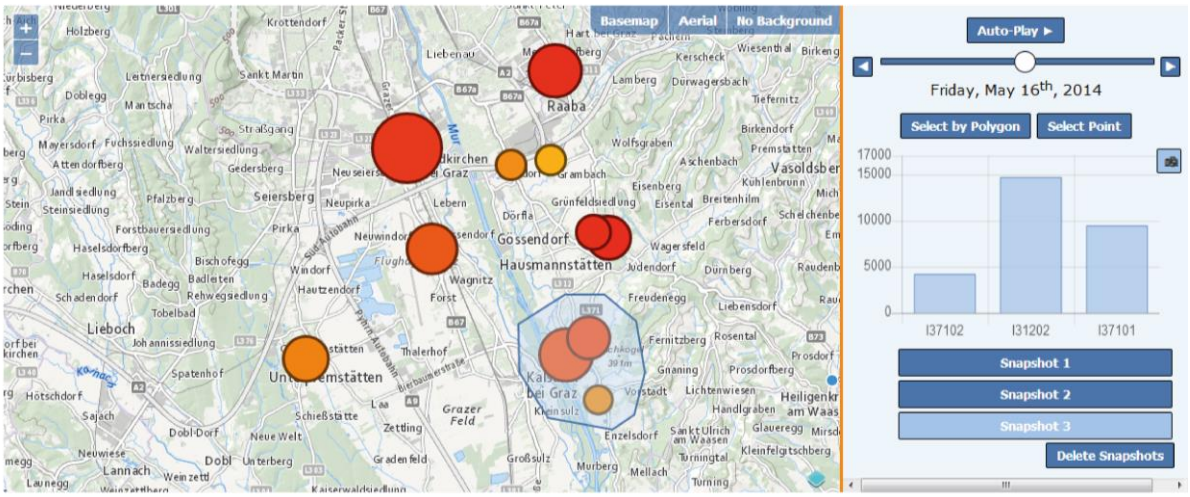


Abbildung 47 Snapshot Buttons zur temporären Speicherung und Wiederherstellung eines Zeitpunktes und Featureselektion. (Quelle: eigene Abbildung)

7 RESÜMEE

Geotemporale Daten sind durch ihre mutlidimensionale Natur nur über Umwege auf zweidimensionalen Oberflächen visualisierbar. Die Möglichkeit des Benutzers, sowohl mit der Karte als auch mit den Daten selbst zu interagieren, stellt einen gravierenden Vorteil von dynamischen, interaktiven Webkarten gegenüber analogen Darstellungen dar.

Mit der Applikation Mapoch wurde erfolgreich ein Programm entwickelt, welches in der Lage ist, geotemporale Daten ohne Serverupload lokal im Browser geeignet zu visualisieren. Hierzu konnte durch den Einsatz offener Standards des W3C und Open Source Software auf proprietäre Software verzichtet werden. Auch Mapoch selbst ist als Open Source Software frei verfügbar, der Code kann online eingesehen und für spezialisierte Anwendungen angepasst werden.

Es wurde große Aufmerksamkeit darauf gelegt, sowohl die Darstellungen als auch die geforderten Datenformate, sollten sie von den offenen Standards abweichen, möglichst offen zu gestalten. Dadurch soll eine heterogene Benutzerbasis erreicht werden, welche mit wenig Mehraufwand vorhandene geotemporale Datensätze visualisieren und mit ihnen interagieren kann.

Dieser offene Ansatz kann andererseits als Kritikpunkt für jene Benutzergruppen aufgefasst werden, die Mapoch häufig mit derselben Art von Daten benutzen möchten. Für solche Power User sollte, mit Mapoch als inhaltliche oder technische Grundlage, eine an ihre Bedürfnisse angepasste Spezialanwendung erstellt werden. Die verwendeten Hintergrundkarten und die Farbgebung der verwendeten Diagramme sind sehr einfach zu verändern, auch die Art der Diagramme und die verlangten Datenformate können mit wenig Programmierkenntnissen individualisiert werden. Ebenso sind fixe Messpunktkoordinaten mit sich verändernden Messdaten denkbar, oder ein automatisiertes Laden der Daten über einen Server.

8 AUSBLICK

Aufgrund des immer stärker wachsenden Bewusstseins gegenüber Datensicherheit und der immer stärkeren Endgeräte, sowohl der Desktops als auch im mobilen Segment, werden sich clientseitige Webapplikationen in den nächsten Jahren weiter verbreiten. Das Verständnis über Internetkriminalität und Computerviren verändert die Hemmschwelle der Benützer gegenüber dem expliziten Downloads von Programmen und Tools, welche Funktionen abdecken sollen, die von den verbreiteten Desktop-GIS nicht abgedeckt werden. Durch diese Faktoren öffnet sich eine Nische für spezialisierte WebGIS-Anwendungen ohne Down- oder Uploads.

Bezüglich der Darstellungsformen ist ein Trend zurück zu klaren und einfachen Grafiken zu erkennen. Das Web- und Datendesign sind in den letzten Jahren erwachsen, mit User Interface Design und User Experience Design konnten sich Disziplinen etablieren, die dediziert die Wahrnehmung des Benutzers in den Vordergrund stellen. Auch in der Webkartographie wird der Benutzer in Zukunft noch genauer analysiert werden müssen, ein interdisziplinarer Ansatz, wie auch hier in dieser Masterarbeit versucht wurde, ist unumgänglich. Abzuwarten bleibt, ob sich 3D-Technologien oder andere technologische Neuerungen in einem relevanten Maß auf das Design von Webkarten und Geodaten auswirken, oder ob die dadurch ermöglichten Darstellungsformen einer akkuraten Informationsübertragung eher entgegenstehen.

Für die Applikation Mapoch ist in Zukunft vor allem eine einheitliche Standardisierung geotemporaler Daten in einem lesbaren Format interessant. Hierzu würde sich besonders das weit verbreitete Austauschformat GeoJSON anbieten. Die Applikation selbst wird idealerweise als Fallstudie beziehungsweise als Vorlage für künftige Spezialapplikationen im Web dienen.

9 LITERATUR

9.1 Buchquellen

ABDALLA, R. (2016): Introduction to Geospatial Information and Communication Technology (GeoICT). Springer International Publishing AG, Schweiz

BARFORD, A. & DORLING, D. (2008): Telling an Old Story with New Maps. In: Dodge, M. et al (Hg.): Geographic Visualization. Concepts, Tools and Applications. Chichester: John Wiley & Sons Ltd, S. 67-107

BURKHARD, R. (2006): Knowledge Visualization: Die nächste Herausforderung für Semantic Web Forschende? In: Pellegrini, T., Blumauer, A. (Hg.): Semantic Web Wege zur vernetzten Wissensgesellschaft. Berlin, Springer-Verlag, Heidelberg, S. 201-212

FEW, S. (2012): Show Me the Numbers – Designing Tables and Graphs to Enlighten. 2. Auflage. Analytics Press, Burlingame CA

FRIES, C. & WITT, R. (2007): Aufs Ganze. Konzeptionelles Gestalten im Zeitalter der Unaufmerksamkeit. Verlag Hermann Schmidt, Mainz

HARROWER, M. & FABRIKANT, S. (2008): The Role of Map Animation for Geographic Visualization. In: Dodge, M. et al (Hg.): Geographic Visualization. Concepts, Tools and Applications. Chichester: John Wiley & Sons Ltd (325). S. 49-65.

KALAWSKY, R. (2009): Gaining Greater Insight Through Interactive Visualization: A Human Factors Perspective. In: Adriaansen, T., Liere, R., Zudilova-Seinstra, E. (Hg.): Trends in Interactive Visualization – State-of-the-Art Survey. Springer Verlag, London, S. 119-154

KORTHAUS, C. (2013): Grundkurs Grafik und Gestaltung. Für Ausbildung und Praxis. Galileo Press, Bonn

KRAAK, M.-J. (2008): Geovisualization and Time – New Opportunities fort the Space-Time Cube. In: Dodge, M. et al (Hg.): Geographic Visualization. Concepts, Tools and Applications. Chichester: John Wiley & Sons Ltd (325). S. 293-306.

MITCHEL, T. (2005): Web Mapping Illustrated. Using Open Source GIS Toolkits, O'Reilly Media, Inc, Sebastopol (CA)

Monmonier, M. (1991): How to Lie with Maps. The University of Chicago Press, Chicago and London

NÖLLENBURG, M. (2006): Geographic Visualization. Conference: Human-Centered Visualization Environments, GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany S. 257-294

NUSSBAUMER KNAFLIC, C. (2015): Storytelling with Data – a data visualization guide for business professionals. John Wiley & Sons, Inc., Hoboken, New Jersey.

RODRIGUEZ, J. & KACZMAREK, P. (2016): Visualizing Financial Data. Indeanapolis: John Wiley & Sons Inc

TRAN, P. V. & NGUYEN, H.T. (2012): Multivariate-Space-Time Cube to Visualize Multivariate Data. In: International Journal of Geoinformatics 8. S. 67-74.

TYNER, JUDITH A. (2010): Principles of Map Design. The Guilford Press, New York

TYNER, JUDITH A. (2015): The World of Maps - Map Reading and Interpretation for the 21st Century. New York

YANG, C. et al (2015): Contempory Computing Technologies for Processing Big Spatiotemporal Data. In: Kwan, M. et al (Hg.): Space-time Integration in Geograpgy and GIScience – Research Frontiers in the US and China. Springer Science+Business Media, Dordrecht, S. 327-351.

ZELAZNY, G. (2001): Say It With Charts – The Executive's Guide to Visual Communication. 4. Auflage. McGraw-Hill Companies

ZIMMERMANN, A. (2012): Basismodelle der Geoinformatik – Strukturen, Algorithmen und Programmierbeispiele in Java. Carl Hanser Verlag, München

9.2 Digitale Quellen

AGAFONKIN, V. (2017): an open-source JavaScript library for mobile-friendly interactive maps. http://leafletjs.com/

BAAJ, A. (2017): Compare the Best Javascript Chart Libraries of 2017. https://blog.sicara.com/compare-best-javascript-chart-libraries-2017-89fbe8cb112d

BABICH, N. (2016): Best Practices For Animated Progress Indicators https://www.smashingmagazine.com/2016/12/best-practices-for-animated-progress-indicators/

BLOCH, M. (2017): Mapshaper Command Reference. https://github.com/mbloch/mapshaper/wiki/Command-Reference

BOSTOCK, M. (2017): D3 Data-Driven Documents. https://d3js.org/

BROADLEY, C. (2018): How do colors work on the web in 2018? https://digital.com/blog/web-colors/

BUNDESMINISTERIUM FÜR INNERES (2018): Nationalratswahlen https://www.bmi.gv.at/412/Nationalratswahlen/Nationalratswahl_2017/

CAMOES, J (2012): Animation, Small Multiples or the Reorderable Matrix? Growth of Walmart, Excel Edition. https://excelcharts.com/animation-small-multiples-growth-walmart-excel-edition/

CARTO (2017): CARTO Editor Workflow. https://carto.com/docs/carto-editor/workflow/

MAPITGIS (2107): MapIt User Guide. http://mapit-gis.com/mapit-user-guide/

MAPITGIS (2107): Export Data. http://mapit-gis.com/export-layer-features/

Chart.js (2017): http://www.chartjs.org/docs/latest/

CHARTBLOCK (2017): When to use a bar Chart. https://www.chartblocks.com/en/support/faqs/faq/when-to-use-a-bar-chart

DUBOST, K. (2016): Use international date format (ISO). http://www.w3.org/QA/Tips/iso-date

EEA (2016): Chart dos and don'ts. https://www.eea.europa.eu/data-and-maps/daviz/learn-more/chart-dos-and-donts

ESRI (2017): WMTS Services. http://server.arcgis.com/en/server/latest/publish-services/linux/wmts-services.htm

ESRI: Shapefile file extensions http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?topicname=Shapefile_file_extensions

FHP (2008): Small Multiples. http://geo.patternbrowser.org/code/pattern/pattern_anzeigen.php?4,298,17,0,0,319

FIELD, K. (2014): Life in Los Angeles by Eugene Turner. http://mapdesign.icaci.org/2014/12/mapcarte-353365-life-in-los-angeles-by-eugene-turner-1977/

FITZJOHN, R. (2013): Excel and line endings https://nicercode.github.io/blog/2013-04-30-excel-and-line-endings/

FULTON, J. (2013): HTML5 Canvas. O'Reilly Media, 2. Edition http://proquest.techbus.safaribooksonline.de/book/web-development/html/9781449335847/1dot-introduction-to-html5-canvas/ch01_html?uicode=tugraz

GILLIES, S. (2014): Pruning CRS from GeoJSON. https://sgillies.net/2014/08/06/pruning-crs-from-geojson.html

HELLER, M. (2017): What is Node.js? The JavaScript runtime explained.

Highcharts (2017): Licenses. https://shop.highsoft.com/highcharts

ISO (2017): ISO/TC211. https://www.iso.org/committee/54904.html

ISO (2017): Members. https://www.iso.org/members.html

KAVUN, S. (2016): L.CanvasIcon https://github.com/sashakavun/leaflet-canvasicon

KLIEVER, J. (2015): Designing With Light and Shadow: 10 Highly Effective Tips You Should Try [With Case Studies] https://designschool.canva.com/blog/light-and-shadow/

KLOKAN TECHNOLOGIES (2016): EPSG.io: Find Coordinate Systems Worldwide. http://epsg.io/about

KOSARA, R. (2010): Understanding Pie Charts. https://eagereyes.org/techniques/Pie Charts

MACWRIGHT, T. (2015): More than you ever wanted to know about GeoJSON. https://macwright.org/2015/03/23/geojson-second-bite.html#projections

MAPBOX (2017): geojson.io. https://github.com/mapbox/geojson.io

MCKENNA, J. (2017): WMS Server. http://mapserver.org/ogc/wms_server.html

MEAGHER, T. (2016): Chernoff Fish https://github.com/tmm/chernoff-fish

MOZILLA AND INDIVIDUAL CONTRIBUTORS (2018): date https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

MOZILLA AND INDIVIDUAL CONTRIBUTORS (2018): Date.prototype.getDay() https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getDay

MOZILLA AND INDIVIDUAL CONTRIBUTORS (2018): FileSystem https://developer.mozilla.org/en-US/docs/Web/API/FileSystem

Mozilla and individual contributors (2018): Map https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

MYGEODATA: https://mygeodata.cloud/converter/shp-to-geojson

OGC (2017): GeoPackage. http://www.geopackage.org/

OGC (2017): KML http://www.opengeospatial.org/standards/kml

OGC (2017): OGC. http://www.opengeospatial.org/ogc

OPEN SOURCE GEOSPATIAL FOUNDATION (2014): What is Geoserver? http://geoserver.org/about/

OPENLAYERS (2017): A high-performance, feature-packed library for all your mapping needs. https://openlayers.org/

OPENLAYERS (2017): ol.source.WMTS https://openlayers.org/en/latest/apidoc/ol.source.WMTS.html

OpenLayers (2017): ol.style.Icon https://openlayers.org/en/latest/apidoc/ol.style.Icon.html

OPENLAYERS (2017): Tile Load Events. https://openlayers.org/en/latest/examples/tile-load-events.html

OPENLAYERS (2018): ol.layer.vector https://openlayers.org/en/latest/apidoc/ol.layer.Vector.html

OSGeo (2017): Geography Markup Language (GML) https://live.osgeo.org/de/standards/gml_overview.html

OSGEO (2017): OpenLayers Info Sheet. http://www.osgeo.org/openlayers

OSGEO (2017): WFS reference. http://geoserver.readthedocs.io/en/latest/services/wfs/reference.html

OSGEO: WMS output formats. http://geoserver.readthedocs.io/en/latest/services/wms/outputformats.html

OSM: German Style. http://wiki.openstreetmap.org/wiki/German_Style

OSM: Standard tile layer. http://wiki.openstreetmap.org/wiki/Standard_tile_layer

PAVLENKO, A. (2017): mapnik. http://mapnik.org/

RODDEN, K. (2017): Sequences sunburst (d3 v4). https://bl.ocks.org/kerryrodden/766f8f6d31f645c39f488a0befa1e3c8

RUDIS, B. (2016): Create Vega-Lite specs & widgets with the vegalite package https://rud.is/b/2016/02/27/create-vega-lite-specs-widgets-with-the-vegalite-package/

SCHAUB, T. (2017): OpenLayers, Rollup, and Closure Compiler. https://bl.ocks.org/tschaub/32a5692bedac5254da24fa3b12072f35

SHAFRANOVICH, Y. (2005): Common Format and MIME Type for Comma-Separated Values (CSV) Files. https://tools.ietf.org/html/rfc4180#page-2

SHAHID, B. (2014): Highcharts Essentials. http://proquest.techbus.safaribooksonline.de/book/databases-and-reporting-tools/9781783983964/1dot-getting-started-with-highcharts/ch01lvl1sec08_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODE3ODM5ODM5NjQlMkZjaDAxbHZsMXNlYzA4X2h0bWwmcXVlcnk9

SINGH, G. (2015): Chernoff Faces with D3 and React.js https://kilotau.com/chernoff-faces-with-d3-and-react-js/

SMUS, B. (2009): Performance of canvas versus SVG. http://smus.com/canvas-vs-svg-performance/

STADT WIEN et al (2016): Willkommen bei basemap.at – der Verwaltungsgrundkarte von Österreich. https://basemap.at/

STAMEN (2018): http://maps.stamen.com/#toner/12/37.7406/-122.3058

STARR, B. (2015): How to Design Bubble Charts. https://visage.co/data-visualization-101-bubble-charts/

THE NEW YORK TIMES COMPANY (2016): Changes in the Electorate in Key States and Regions. http://www.nytimes.com/interactive/2012/11/11/sunday-review/counties-moving.html

W3.org (2011): HTML/Canvas and SVG. https://www.w3.org/community/webed/wiki/HTML/Canvas_and_SVG

W3CSchools (2017): HTML5 SVG. https://www.w3schools.com/html/html5_svg.asp

WEBPACK (2017): Tree Shaking. https://webpack.js.org/guides/tree-shaking/

WILSON, T. (2007): KML 2.2 – An OGC Best Practice. https://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/%3fartifact_id=23689