Informationsvisualisierung und Visual Analytics

Zusammenfassung

Fabian Damken

12. März 2022

TECHNISCHE UNIVERSITÄT DARMSTADT

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

1 Einleitung

In dieser Zusammenfassung werden zwei Themen behandelt: Informationsvisualisierung und Visual Analytics. Dabei sollen die Frage beantwortet werden, wie verschiedene Daten *gut* visualisiert werden können und wie Visualisierung die Analyse unterstützen können. Viele Ideen der folgenden Kapitel und grundlegender Techniken bauen dabei auf dem Verständnis grundlegender Prozesse des Gehirns ab. Denn: Eine Visualisierung soll dem Gehirn des*der Nutzer*in Arbeit abnehmen. Dabei sollen für jede Visualisierung die folgenden drei Fragen beantwortet werden:

- Was wird wie dargestellt?: Formale Beschreibung einer Visualisierung.
- Was ist gut und (möglichst) ohne Anstrengung sichtbar?: Prinzipien der Wahrnehmung kennen und anwenden.
- Was hilft dem*der Nutzer*in bei der Aufgabe?: Beschreibung und Wahrnehmungsprinzipien im Kontext einer Aufgabe bewerten.

Ein relevantes, bisher noch nicht erwähntes, Wort in den obigen Fragen ist die *Aufgabe*. Zu Beginn jeder Visualisierung muss zunächst die *Aufgabe* der Visualisierung identifiziert werden. Dies sind oftmals Entscheidungen, die (objektiv) durch Daten und Informationen getroffen werden (sollten). Dies ist in Abbildung 1.1 dargestellt. In der Praxis werden jedoch häufig einfach Visualisierungkataloge (große Datenbanken mit Visualisierungstechniken) nach einer „schönen“ Visualisierung durchsucht. Dadurch wird das Dasein der Visualisierung als Werkzeug jedoch zu dem Zweck gemacht, d. h. die Visualisierung wird ein Selbstzweck. Dies sollte eigentlich nie der Fall sein!

Oft die Aussage getroffen, dass „ein Bild mehr sagt als tausend Worte.“ Im Allgemeinen sollte allerdings eher gesagt werden, dass ein Bild etwas *anderes* als tausend Worte sagt: Sprachliche Artefakte (wozu auch Zahlen gehören), werden von dem Gehirn *bewusst* und verarbeitet und oftmals in eine zeitliche Reihenfolge gebracht. Eine Visualisierung der selben Daten hingegen ist ein bildliches Artefakt und erlaubt eine *unbewusste* Verarbeitung, bei der die Informationen im Raum strukturiert werden. Dadurch können komplexe Zusammenhänge sehr schnell vermittelt und erfasst werden.

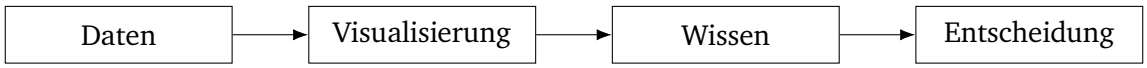

Abbildung 1.1: Eine Visualisierung dient immer der Erfüllung einer Aufgabe und soll zu einem Erkenntnisgewinn führen. Oftmals steht am Ende davon eine Entscheidung, die objektiv durch Daten getroffen werden soll. Bei der Erstellung einer Visualisierung sollte dieser Prozess also rückwärts durchgeführt werden, d. h. ausgehend von der Frage, welche Entscheidung getroffen werden soll.

1.1 Anwendungen von Visualisierungen

Die Anwendung einer Visualisierung, lässt sich in zwei Kategorien einteilen: *Erfassen* und *Produzieren* von Informationen, wobei erstere durch Informationsvisualisierung und letztere durch Visual Analytics „bearbeitet“ werden.

Innerhalb der Informationsvisualisierung werden die folgenden Gruppen unterschieden:

- *Explain:* Es sollen bekannte Informationen an andere vermittelt werden (möglicherweise, aber nicht immer, interaktiv).
- *Explore:* Es sollen neue Information auf Basis von Daten gefunden oder unsichere Informationen bestätigt werden (üblicherweise sehr interaktiv; das Ziel ist nicht immer bekannt).
- *Enjoy:* Zwanglose und durch Neugier getriebene Begegnung mit den Daten; dabei ist die Aufgabe selten bekannt.

Von diesen drei Arten der Informationsvisualisierung werden in dieser Zusammenfassung vor allem die ersten beiden behandelt.

Innerhalb der Visual Analytics werden die folgenden Gruppen unterschieden:

- *Annotate*
- *Record*
- *Derive*

1.2 Identifizierung der Visualisierungsaufgabe

Bei der Identifizierung der Aufgabe sollte auch mit einbezogen werden,

- welche Informationen als bekannt vorausgesetzt werden,
- welche Informationen gesucht werden, und
- was mit den neuen Informationen gemacht wird oder gemacht werden soll.

Das Design der Visualisierung bestimmt damit essentiell, wie gut mit der Visualisierung gearbeitet werden kann durch Wiedererkennung bekannter Informationen und Erkennung neuer Informationen. Die erste Regel ist dabei, wie bereits erwähnt, das die Visualisierung ein Werkzeug und kein Selbstzweck ist. Das lässt sich wie folgt zusammenfassen:

Have something to tell or something to ask!

1.3 Negativbeispiele

2 Der Informationsvisualisierungsprozess

Der *Informationsvisualisierungsprozess*, dargestellt in Abbildung 2.1, wurde von Card et al. als Referenzmodell entworfen, um die Erstellung einer Visualisierung zu strukturieren. Dabei wird zwischen den Repräsentationen (*Rohdaten*, *Datentabellen*, *Visuelle Strukturen* und *Views*) unterschieden. Der Übergang zwischen den Strukturen wird durch die *Datentransformation*, die *Visuelle Abbildung* und die *View Transformation* beschrieben. Eine weitere Zentrale Komponente ist der*die Nutzer*in (*Mensch*), welche*r die einzelnen Schritte modifiziert. Dadurch wird das Modell zu einem Kreislauf der Visualisierung. In diesem Kapitel wird jeder Schritt separat behandelt, beginnend mit der Datentransformation (Abschnitt 2.1) über die visuelle Abbildung (Abschnitt 2.2) bis zur Interaktion (Abschnitt 2.4). Im folgenden wird eine kurze Übersicht über die Transformationsschritte gegebene:

- *Datentransformation:* Der der *Datentransformation* werden die *Rohdaten* in *Datentabellen* umgewandelt. Diese können dann in den nächsten Schritten einfacher als die Rohdaten verarbeitet werden, da diese häufig ungeordnet und unstrukturiert vorliegen.
- *Visuelle Abbildung:* Bei der *visuellen Abbildung* findet der Hauptteil der Visualisierung statt. Dabei werden einzelne Datenvariablen auf visuelle Attribute (bspw. die Position) abgebildet, was die *visuellen Strukturen*.
- *Datentransformation:* Abschließend werden in der *View Transformationen* die tatsächlichen Datenwerte auf die Variablenwerte der visuellen Struktur abgebildet, woraus sich die tatsächliche Visualisierung, die *View*, ergibt.

Die Unterscheidung zwischen der visuellen Abbildung und der View Transformation ist, dass ersteres beschreibt, *dass* eine Datenvariable auf eine visuelle Variable abgebildet wird, und letzteres beschreibt, *welcher* konkrete Datenpunkt auf welchen konkreten Wert abgebildet wird.

Der letzte Schritt im Informationsvisualisierungsprozess ist die Interaktion, welche durch die zurückgeführten Pfeile in Abbildung 2.1 dargestellt wird. Dies ist die technische Möglichkeit, die einzelnen Schritte zu verändern und anzupassen. Es soll dem*der Nutzer*in dabei möglich sein, einzelne Komponenten anzupassen, ohne auf eine neue Visualisierung des*der Designer*in zu warten. Für den*die Designer*in kann dadurch zu teilen das Problem gelöst werden, dass eventuell die genaue Aufgabe noch nicht bekannt ist.

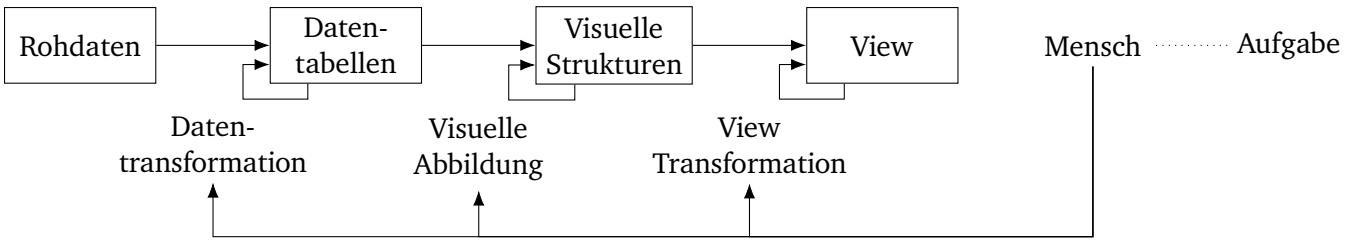

Abbildung 2.1: Der Informationsvisualisierungsprozess nach Card et al.

2.1 Die Datentransformation

Bei der Datentransformation werden die Rohdaten in Datentabellen umgewandelt, welche in den folgenden Schritten einfacher zu verarbeiten sind. Außerdem können eine Reihe an Vorverarbeitungstechniken (bspw. Datenbereinigung) angewandt werden. Dieser Abschnitt führt zunächst in die grundlegenden Datentypen und -strukturen sowie anschließend verschiedene Techniken der Vorverarbeitung ein.

2.1.1 Datentypen und Datenstrukturen

Zur Diskussion und Abbildung von Daten auf visuelle Strukturen ist es zunächst sinnvoll, die verschiedenen auftretenden Datentypen und -strukturen zu betrachten. Die Unterscheidung zwischen einem Typ und einer Struktur wird dabei grundlegend durch die Anzahl der zugeordneten Datenvariablen getroffen: Einfache Datentypen beziehen sich immer auf genau eine Datenvariable während Datenstrukturen Beziehungen zwischen mehreren Datenvariablen (welche meist Eigenschaften von Objekten, auch *Items* genannt, sind) und Datensätzen beschreiben. Eine Visualisierung kann anschließend eine beliebige Kombination aus Datenvariablen, Items und Beziehungen abbilden.

Datentypen

Grundlegend wird zwischen *nominalen*, *ordinalen* und *quantitativen* Datentypen unterschieden. Bei einem nominalen Datentyp ist dabei zwischen zwei Werten nur *Gleichheit* und *Ungleichheit* definiert. Dies können z. B. Personen, Länder, etc. sein. Sind nur nominelle Daten zu visualisieren, kann eine Tabelle hier tatsächlich die beste Wahl sein! Ordinale Datentypen haben zusätzlich zu (Un-) Gleichheit noch eine *Ordnung*, bspw. Schulnoten oder Rangordnungen. Der in Anzahl Operationen „mächtigste“ Datentyp sind quantitative Daten, die auch noch *Differenzen* und *Differenzverhältnisse* haben. Ferner wird bei quantitativen Daten zwischen *diskreten* und *kontinuierlichen* Daten. Außerdem gibt es *Intervall-* und *Verhältnisskalen*: Auf ersterer sind Verhältnisse nicht vernünftig berechenbar, da die Skala keinen natürlichen Nullpunkt hat (bspw. Temperatur in °C). Letztere hat einen natürlichen Nullpunkt, weshalb auch Verhältnisse auf Werten Sinn ergeben (bspw. Temperatur in Kelvin). Des weiteren gibt es *lineare* und *zyklische* Skalen, bei der die Ordnung eine Totalordnung, bzw. eine „künstliche“ Ordnung, ist.

Die unterschiedlichen Datentypen sind in Abbildung 2.2 zusammengefasst. Im Allgemeinen lässt sich also sagen, dass sich Datentypen vor allem durch die auf ihnen ausführbaren Operatoren unterscheiden. Außerdem ist nicht die Repräsentation, sondern die Bedeutung des Datentyps relevant (bspw. sehen Schulnoten aus wie Zahlen, sind aber ordinale Daten).

Datenstrukturen

Die bisher diskutierten Daten*typen* beziehen sich immer auf genau eine Datenvariable, wohingegen Daten*strukturen* auch Zusammenhänge zwischen Daten abbilden. Dabei sind Datenvariablen oft Eigenschaften von Objekten, die hier *Item* genannt werden.

Die einfachste Form einer Datenstruktur ist eine Tabelle, die außerdem sehr flexibel ist, weshalb sie im Informationsvisualisierungsprozess (Abbildung 2.1) namensgebend ist. Innerhalb einer Tabelle hat jede Spalte eine von zwei Rollen: Key oder Value. Dabei definieren die Datentype der Keys die eigentliche Datenstruktur und die Values sind von den Keys abhängig. Dabei müssen die Keys eine Zeile eindeutig identifizieren, d. h. es darf keine Kombination mehrfach auftreten.

Hat eine Tabelle keine Keys sondern ausschließlich Attribute, so werden die Items nur technisch Unterschieden (bspw. durch eine Zeilennummer) und die Daten sind „rein“ uni-/bi-/multivariat. Dies ist oft der Fall bei

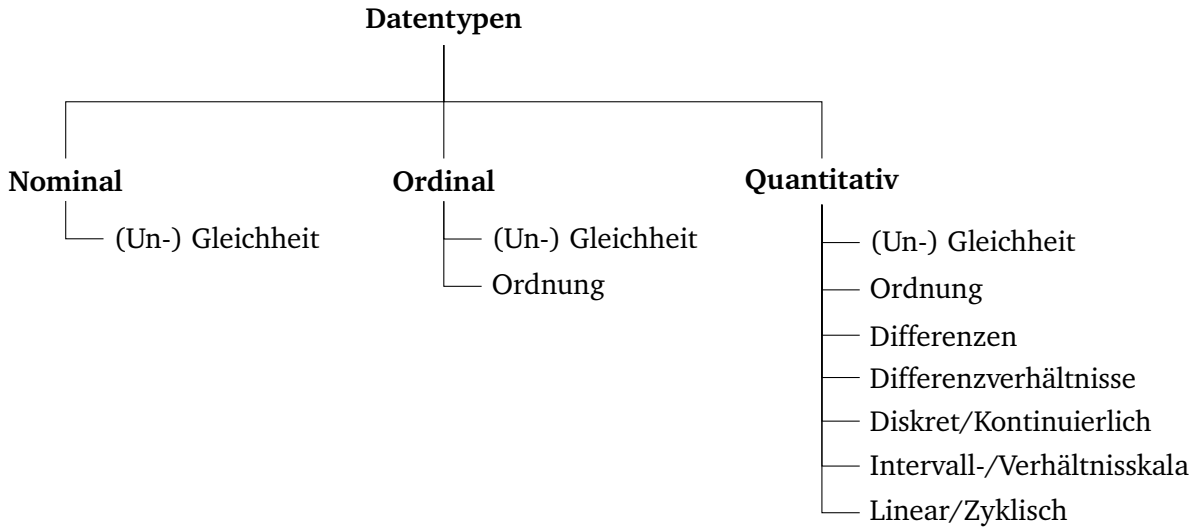

Abbildung 2.2: Unterschiedliche Datentypen

Daten, bei denen die Eigenschaften wichtiger sind als die Identifikation oder bei Daten, in denen die Items nicht identifiziert werden dürfen (Anonymisierung).

Da eine Tabelle sehr flexibel ist, sind die folgenden Datenstrukturen allesamt auf eine Tabelle abbildbar (die Abbildung ist jeweils durch die Angabe der Keys gegeben). Darüber hinausgehend gibt es auch Datenstrukturen, die am besten auf mehrere Tabellen (als relationales Modell) abgebildet werden. Achtung: Zwar ist die Darstellung als Tabelle hier zweckmäßig, jedoch müssen die Daten nicht zwangsweise technisch als Tabelle gespeichert werden!

Zeitbezogene Daten Zeitbezogene Daten haben mindestens einen Zeitstempel als Key, welcher sowohl relativ als auch absolut sein kann. Die Zeit wird häufig über die Position Achse visualisiert (z. B. Corona-Fallzahlen). Die Darstellung zeitbezogener Daten wird intensiv in Abschnitt 3.3 behandelt.

Ortsbezogene Daten Bei ortsbezogenen Daten ist mindestens ein Key eine Ortsangabe, die beispielsweise als geographische Position (Längen- und Breitengrad) oder Ortsname realisiert werden kann. Eine beliebte Darstellung ist das Anzeigen auf einer Karte.

Bewegungsdaten Bewegungsdaten stellen eine Kombination aus zeit- und ortsbezogenen Daten dar. Auch hierbei gibt es mehrere Varianten, bspw. können Messungen an mehreren, aber individuell festen Orten, durchgeführt werden (z. B. Wetterstationen) oder die Messungsorte können sich bewegen (z. B. Sportdaten). Eine beliebte Darstellung dieser Daten ist eine Karte, wahlweise als 3D-Darstellung mit der Höhe als Zeit. Dadurch sind schnelle und langsame Bewegungen unterscheidbar und Kreuzungen im Raum entsprechen Treffpunkten.

Graphen und Netzwerke Graphen und Netzwerke sind spezielle komplexe Datenstrukturen, wobei die Kanten durch zwei Keys desselben Typs auf eine Tabelle abgebildet werden können (die Values sind dann Eigenschaften der Kante, nicht der Knoten). Die Darstellung solcher Daten wird intensiv in Abschnitt 3.4 behandelt.

Bäume und Hierarchien Ein Baum kann als Spezialfall eines Graphs behandelt werden, wobei ein Value-Attribut den Key eines anderen Eintrags enthält, wodurch eine Kindheits-Relation induziert wird. Alternativ wäre eine Darstellung als Kantenliste, analog zu einem allgemeinen Graph, möglich. Im Vergleich zu allgemeinen Graphen gibt es für Bäume viele spezialisierte Visualisierungswerkzeuge, die in Abschnitt 3.4 behandelt werden. Dies geschieht durch die Ausnutzung der Ordnungsrelation, die durch einen Baum definiert wird.

2.1.2 Vorverarbeitung

Ein wichtiger Teil der Datentransformation stell die Datenvorverarbeitung dar[1]. Da reale Daten oft „unsauber" sind, müssen diese bereinigt werden, um dem „Garbage In, Garbage Out"-Prinzip zu entgehen. *Unsauber* hat dabei viele Facetten, z. B. unvollständige und fehlende Werte, Rauschen, Inkonsistenz, Ausreißer und unglaubwürdiger Werte, und viele mehr. Die Vorverarbeitung umfasst dabei alle notwendigen Transformationen, die nicht von dem*der Nutzer*in durchgeführt werden können oder sollen. Wie immer ist die genaue Abgrenzung jedoch Abhängig von der Aufgabe und dem*der Nutzer*in: Ist das Fehlen von Werten beispielsweise keine wertvolle Information für den*die Nutzer*in, so sollten diese in der Vorverarbeitung von dem*der Entwickler*in bereinigt werden. Ein anderes Beispiel sind Ausreißer oder unplausible Werte: Ist der*die Entwickler*in bei der Interpretation der Daten überfordert, sollte die Verarbeitung als Datentransformation interaktiv durch den*die Nutzer*in durchführbar sein. Der Einfachheit halber wird deshalb im folgenden immer von (Daten-) Vorverarbeitung gesprochen, wobei jeder Schritt auch als interaktive Transformation dargestellt werden könnte.

Bei der Vorverarbeitung sind insbesondere Metadaten (z. B. Attributnamen, Referenzpunkte von Messungen, Einheiten, wichtige Schlüsselwörter) hilfreich. Außerdem werden häufig statistische Methoden wie Ausreißererkennung, Clusteranalyse, Korrelationsanalyse sowie statistische Plots und Histogramme verwendet.

Fehlende Werte und Datenbereinigung

Um fehlende Werte in den Daten zu korrigieren gibt es einige grundlegende Verfahren:

- *Ignorieren:* Die fehlenden Werte werden ignoriert und es wird gehofft, dass dies in der Visualisierung nicht tiefgreifend durchschlägt.
- *Manuell Einfügen:* Unter Nutzung von Expert*innenwissen werden die Daten manuell ergänzt. Dies ist zwar präzise, aber sehr zeitaufwendig.
- *Eliminieren der Zeile:* Zeilen mit fehlenden Werten werden entfernt. Dies ist die häufigste Methode, jedoch fehlt bei vielen Datensätzen in jeder Zeile mindestens ein Wert.
- *Globale Konstante:* Anstelle der fehlenden Werte wird eine globale Konstante, z. B. -1, eingesetzt. Dies verändert allerdings die Datenverteilung, was Probleme bei statistischen Analysen hervorrufen kann.
- *Mittelwert:* Die Werte werden durch den Mittelwert ersetzt. Dies ist gut für eine globale Statistik, die individuellen Abweichungen können allerdings groß sein.
- *Wert basierenden auf Ähnlichkeit:* Ersetzung der Werte durch Annahme, welche anderen Zeilen „ähnlich" sind.

[1] Nach einer Studie von CrowdFlower im Jahr 2016 entfällt $80\,\%$ der Zeit bei der Erstellung einer Visualisierung auf die Datensuche und -vorverarbeitung und nur etwas $20\,\%$ auf die eigentliche Visualisierung.

Die konkrete Wahl einer dieser Methoden hängt dabei – wie üblich – stark von dem vorliegenden Problem ab, z. B. des Typs und der Semantik der Daten, der Menge der fehlenden Werte und der Expertise des*der Anwenders*Anwenderin. Im Allgemeinen sollte immer versucht werden, den Grund für das Fehlen zu ermitteln. Außerdem muss bei einem Ersetzen der fehlenden Werte gespeichert werden, dass es sich um Ersatzwerte handelt!

Im falle von fehlerhaften Werten, die häufig durch Menschen selbst verursacht werden, können ähnliche Methoden eingesetzt werden, allerdings sind diese sehr viel schwer zu detektieren.

Ausreißer (-detektion)

Ausreißer sind Datenpunkte, die außerhalb des normalen Wertebereichs einer Datenvariable liegen. *Starke* Ausreißer sind solche, die für Expert*innen ungewöhnlich und interessant sind. Dabei ist es nicht immer einfach zu entscheiden, was ein interessanter – aber plausibler – Ausreißer und was ein Fehler ist. Zur Detektion von Ausreißern werden deshalb häufig statistische Verfahren herangezogen. Eine einfache Methode ist die Annahme einer Normalverteilung und Ausschluss aller Werte, die um mehr als zwei Standardabweichungen vom Mittelwert abweichen. Das offensichtliche Problem dieser Annahme ist, dass sie häufig nicht gilt (bspw. da die Daten von mehreren Normalverteilungen stammen; dies kann durch Clustering detektiert werden, was später behandelt wird). Allerdings können, wie bereits erwähnt, Ausreißer auch plausibel und somit keine Datenfehler sein (ein Beispiel hierfür ist die Temperatur auf der Zugspitze, die plausibel einen Ausreißer im Vergleich zu meeresspiegelnahen Messstation darstellt).

Normalisierung und Skalierung

Oftmals würde ein einfaches Anzeigen der Daten dazu führen, dass viele der Datenpunkte an einem Ende der Skala „kleben.“ Hier kann eine Skalierung der Daten abhelfen, indem die Daten auf irgendeine Weise gestreckt/gestaucht werden. Dabei werden die Daten auf ein „normales“ Intervall, z. B. $[0, 1]$ oder $[-1, 1]$, abgebildet. Gängige Methoden sind:

- Min-Max-Normalisierung
  - Linear
  - Logarithmisch
  - Wurzel
  - Quadratisch
  - Exponentiell
- Weitere datenspezifische Normalisierungen (bspw. nach der Objektform)
- z-Score
- Quantil-Normalisierung

Bei der Min-Max-Normalisierung wird das Minimum ℓ und das Maximum u der Daten verwendet, um die restlichen Daten zu skalieren. Zur Abbildung auf das Intervall $[0, 1]$ gibt es z. B. die folgenden Methoden[2]:

$$f_{\text{Linear}}(x) = \frac{x - \ell}{u - \ell} \qquad f_{\text{Log}}(x) = \frac{\log x - \log \ell}{\log u - \log \ell} \qquad f_{\text{Quad}}(x) = \big(f_{\text{Linear}}(x)\big)^2 \qquad f_{\text{Wurzel}} = \sqrt{f_{\text{Linear}}(x)}$$

[2] Zur Interpretation der logarithmischen Normalisierung kann es hilfreich sein, diese etwas umzuformen: $(\log x - \log \ell)/(\log u - \log \ell) = \log(x/\ell)/\log(u/\ell) = \log_{u/\ell}(x/\ell) = \log_{u/\ell} x + \text{const}_x$

Für eine Normalisierung in den Bereich $[-1, 1]$, was für negative, „bipolare“, Werte sinnvoll ist, kann z. B. wieder eine Min-Max-Normalisierung

$$\hat{f}_{\text{Linear}}(x) = 2f_{\text{Linear}})(x) - 1$$

eingesetzt werden. Dabei kann es hilfreich sein, ℓ und u als die absoluten Minima und Maxima zu definieren, d. h.

$$\ell' = -\max\{|\ell|, u\} \qquad\qquad u' = \max\{|\ell|, u\},$$

um sicherzustellen, dass $\hat{f}_{\text{Linear}}(0) = 0$ gilt (dies kann sonst zu einer Verzerrung führen, bspw. bei Temperaturen).

Lokale und Globale Skalierung Gibt es mehrere Datensätze mit unterschiedlichen Minima/Maxima (z. B. Temperaturverlauf in mehreren Städten), so muss entschieden werden, welche Minima/Maxima verwendet werden. Dabei wird zwischen *lokaler* und *globaler* Normalisierung unterschieden: Bei der lokalen Normalisierung werden die Minima/Maxima jeden Datensatz bestimmt und angewendet, bei der globalen Normalisierung wird das Minimum/Maximum über alle Datensätze hinweg verwendet.

Dabei gilt, wie so oft, dass die konkrete Wahl auch von dem konkreten Problem abhängt. Während bei einer lokalen Normalisierung die Wertverläufe gut verglichen werden können (d. h. wann unterschiedliche Reihen z. B. ihr Maximum erreichen), so können dabei die Werte nicht mehr direkt verglichen werden. Komplementär dazu können die Werte bei einer globalen Normalisierung gut verglichen werden, bei großen Unterschieden ist ein Verlaufsvergleich jedoch sehr schwer.

Diskretisierung

Werden kontinuierliche Daten verarbeitet, so ist es häufig nötig, diese zu diskretisieren, d. h. diskrete Teilmengen als Approximation zu finden. Der Hauptgrund dafür ist, dass ein digitales System nicht mit „echt“-kontinuierlichen Daten umgehen kann, da sie inhärent diskret sind. Die üblichste Diskretisierungsdimension ist dabei die Zeit, bei der ein beliebiger Punkt $t_k = k\Delta_t$ für einen Zeitindex k mit vorgeschriebener Schrittweite Δ_t diskretisiert wird. Bei der Diskretisierung gehen allerdings Informationen verloren, was eventuell problematisch werden kann (vgl. Nyquist-Shannon-Abtasttheorem).

Sampling, Segmentierung und Untermengen

Um die Datenmenge zu reduzieren, gibt es verschiedene Ansätze. Eine davon ist *Sampling*, wobei zufällige Stichproben aus den Daten gezogen werden, um die Datenpunkte in der Visualisierung zu reduzieren. Je nach Wahl der Verteilungsfunktion kann sich dadurch allerdings eine Verzerrung in den Daten und einiger statistischer Eigenschaften ergeben. Andere, nicht-zufalls-basierte, Methoden sind *Segmentierung* und die Wahl von *Untermengen*. Bei der Segmentierung werden die Daten in zusammenhängende Abschnitte/Regionen eingeteilt und einer Kategorie zugeordnet. Dies ist allerdings nicht immer eindeutig möglich. Die Wahl von Untermengen kann bspw. auf Basis von Filtern und anderen Einschränkungen erfolgen.

Datenintegration

Bei der *Datenintegration* werden verschiedene Datenquellen zu einer einzigen Datenquelle vereint, was eventuell die Angleichung von Schemata und ähnlichem erfordert. Beispielsweise müssen bei der Integration imperialistischer und metrischer Quellen die Einheiten angepasst werden.

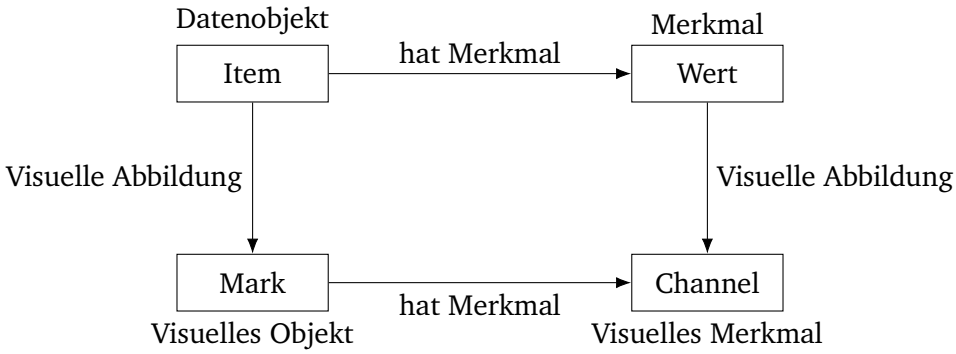

Abbildung 2.3: Beziehung zwischen Items, Werte, visuellen Channels und Marks

2.2 Die Visuelle Abbildung

In diesem Abschnitt wird die visuelle Abbildung, der dritte Schritt im Visualisierungsprozess (Abbildung 2.1) genauer behandelt. Dabei ist die Abgrenzung zum vierten Schritt, der View Transformation, nicht immer scharf gezogen, da die Schritte eng zusammenhängen. In der Regel wird in der visuellen Abbildung eine Datenvariable auf eine visuelle Struktur abgebildet. Dies muss aber nicht immer exakt sein, d. h. eine Variable kann auch auf mehrere Strukturen oder mehrere Variablen können auf die gleiche Struktur abgebildet werden (bspw. wird in einer Scatterplot-Matrix sowohl ein Datenwert als auch eine Dimension auf die Position abgebildet). Im Folgenden werden einige visuelle Strukturen („Marks“ und „Channels“) eingeführt und ihre typische Anwendung beschrieben.

Eine visuelle Abbildung muss immer die folgenden Dinge beinhalten: Titel, Skala und Achsenbeschriftungen (gegebenenfalls mit Einheit), Datenquelle, Legende und den Grafikkörper selbst. Außerdem sollte üblicherweise eine Bildunterschrift mit angegeben werden, siehe dazu **??**.

2.2.1 Beispiele

2.2.2 Visuelle Strukturen

In diesem Abschnitt werden eine visuelle Strukturen – auch *Marks* und *Channels* genannt – eingeführt und ihre übliche Einsetzung beschrieben. Dabei beginnt jede Visualisierung grundlegend mit einem leeren Blatt, dem die visuellen Strukturen hinzugefügt werden.

Der *Raum*, d. h. die Position auf der Abbildung, ist die bei weitem wichtigste und vielseitigste Struktur. Sofern nicht anders angegeben beziehen sich die Positionen immer auf zwei unabhängige[3] Variablen (x und y). Die Position ist dabei auch die einzige Struktur, die unabhängig von der Aufgabe und dem Datentyp verwendet werden kann und – ebenfalls als einzige Struktur – verwendet werden muss. Daher ist die Belegung der visuellen Raumvariablen die erste und wichtigste Designentscheidung. Innerhalb des Raumes werden anschließend weitere visuelle Strukturen platziert, die in Marks und Channels eingeteilt sind: *Marks* sind geometrische Elemente, aus denen die Visualisierung zusammengesetzt sind, während *Channels* Eigenschaften der Markierungen/Marks sind.

Die visuelle Abbildungen besteht abschließend aus einer Zuordnung von Datenvariablen und Merkmalen auf Marks und Channels, siehe Abbildung 2.3.

[3] Die Wortwahl *unabhängig* bezieht sich hier nur darauf, dass die Variablen selbst, nicht zwangsweise die Werte, unabhängig sind.

Punkt	Linie	Flächen
Position	Position	Position
Farbe	Farbe	Farbe
Helligkeit	Helligkeit	Helligkeit
Form	Breite	Kanteneigenschaften (siehe Linie)
Orientierung	Stil (Gestrichelt, Gepunktet, ...)	Tiefe
Größe	Größe/Länge	Größe/Fläche
Textur	Dynamik	Textur

Tabelle 2.1: Übliche Channels

Marks

Marks sind die eigentlichen Strukturelemente, die auf der Visualisierung abgebildet werden, die häufig jeweils ein Item oder eine Relation repräsentieren. Die üblichsten sind dabei *Punkte*, *Linien*, *Flächen* und *Text* wobei letzterer nur sehr sparsam und am besten gar nicht eingesetzt werden sollte. Zur Auswahl eines Marks sollten mehrere Aspekte in Betracht gezogen werden, insbesondere die Daten, die repräsentiert werden sollen (einzelne Items, Item-Paare, Teilmengen von Items, ein Attribut, etc.). Zum Verständnis der Visualisierung kann es hilfreich sein, wenn eine eingängige Metapher zwischen den repräsentierten Daten und der Visualisierung besteht (z. B. Darstellung eines Volumens durch eine Fläche). Dabei stellen Linien häufig dar, dass ein Änderung kontinuierlich verläuft, während Punktmengen und ähnliches andeuten, dass eine Änderung diskret ist. Des weiteren sollte bei der Auswahl der Marks darauf geachtet werden, dass die genutzten Marks die benötigten Channels (siehe nächster Abschnitt) unterstützen.

Eine Ambiguität zwischen den obigen Marks ist die Unterscheidung zwischen einem Punkt und einer Fläche: Wann ist ein Punkt „so groß“, dass er eigentlich eine Fläche ist? Dies kann dadurch aufgelöst werden, dass ein Punkt allgemein eine Markierung bezeichnet, die etwas 0-Dimensionales darstellt, d. h. auch ein großer Punkt markiert ausschließlich einen Punkt, indem der Bezug zum Hintergrund verschieden ist.

Channels

Channels sind die visuellen Eigenschaften von Marks, d. h. diese können zusätzlich als Unterscheidungs- und Identifikationsmerkmal eingesetzt werden. Dabei unterscheiden sich die anwendbaren Channels nach den eingesetzten Marks. Eine Übersicht über die üblichsten Channels ist in Tabelle 2.1 zu finden. In diesem Abschnitt werden diese üblichen Channels sowie einige nicht häufig anzutreffende Channels vorgestellt. Ebenfalls wird auf die Rolle der Elemente im Designprozess eingegangen, wobei der Grund für diese Rolle in Abschnitt 2.3 behandelt wird. Eine Übersicht über die Eignung der gängigsten Channels für verschiedene Datentypen und Aufgaben ist in Tabelle 2.2 bzw. Tabelle 2.3 gegeben.

Im allgemeinen sollten nicht zu viele Channels verwendet werden, da die Abbildung dadurch chaotisch wird. Im konkreten Fall hängt die genaue Anzahl natürlich immer von der Aufgabe ab. Für Explain- und Explore-Aufgaben sollten generell eher weniger Channels verwendet werden, damit die Visualisierung verständlich bleibt. Außerdem sollte bei „Explore“ vermieden werden, dass dominante Channels die Wahrnehmung von Mustern in anderen Channels verhindern. Bei „Enjoy“ hingegen ist absolute Freiheit gegeben.

Farbkanäle (Farbton, Helligkeit, Sättigung) Ein sehr häufiger Channel ist die Farbe von Marks, die gleich drei verschiedene Channels aufweist: Farbton, Helligkeit und Sättigung, wobei allerdings Helligkeit und Sättigung praktisch niemals zeitgleich genutzt werden. Die Farbe kann dabei viele verschiedene Werte diskreter und

	Nominal	Ordinal	Quantitativ	Räumlich	Zeitlich
Position	+	+	+	+	+
Länge	−	+	+	?	?
Größe	−	+	○	−	−
Farbsättigung	−	+	○	−	−
Textur	+	+	−	−	−
Farbton	+	(−)	−	−	−
Orientierung	+	+	−	−	−
Form	+	−	−	−	−

Tabelle 2.2: Übliche Channels und deren Eignung für verschiedene Datentypen

	Grupp.	Sel./Hervorhebung	Vgl. Anord.	Vgl. Quant.	Unterscheidbarer Werte
Position	+	+	+	+	Bildschirmgröße
Länge	−	(+)	+	+	ca. 5 bis 15
Größe	−	+	+	+	ca. 5 bis 15
Farbsättigung	−	+	+	+	ca. 5 bis 7
Textur	+	+	+/−	+/−	ca. 5 bis 7
Farbton	+	+	−	−	ca. 7 bis 8
Orientierung	+	+	○	○	ca. 4 bis 6
Form	+	○	−	−	ca. 5 bis 7 „neutrale“

Tabelle 2.3: Übliche Channels und deren Eignung für verschiedene Aufgaben. Abkürzungen: „Grupp.“ ist „Gruppierung“, „Sel.“ ist „Selektion“, „Vgl.“ ist „Vergleich“, „Anord.“ ist „Anordnung“ und „Quant.“ ist „Quantitäten“.

kontinuierlicher Natur darstellen und ist die einzige Struktur, die auch bei nur einem Pixel funktioniert. Die Nutzbarkeit und Barrierefreiheit ist jedoch stark von der gewählten Farbskala abhängig (dies wird in Abschnitt 2.3.1 genauer behandelt).

Länge Neben der Position ist die *Länge* eines Objekts das einzige Attribut, welches numerische Größenverhältnisse exakt abbilden kann (im Falle von parallelen Längen sogar besser als die Position). Jedoch ist die Wahrnehmung stark beeinflussbar, weshalb die Darstellung sehr einfach sein muss, wenn tatsächlich numerische Vergleiche durchgeführt werden sollen, d. h. alles außer Länge und Position sollte weggelassen werden (Balkendiagramme).

Größe und Flächeninhalt Für einen qualitativen Vergleich ist es naheliegend, Größenordnungen zu vergleichen. Insbesondere bei einer Fläche sind *Differenzen* allerdings sehr schwer vergleichbar, da die Fläche quadratisch mit der Breite skaliert. Des weiteren sind Länge und Breite nicht als separate Channels zu betrachten, da sich dadurch eine Fläche ergibt, deren Größe eher wahrgenommen wird. Dadurch können die eigentlichen Daten überdeckt werden. Ein weiterer Nachteil ist, dass die Größe als Platz benötigt.

Form Die *Form* kann als Channel sehr gut gelernt werden, d. h. die Formen erhalten über die Zeit eine feste Bedeutung für den*die Nutzer*in und sie werden wiedererkannt. Da es sehr viele vorstellbare Formen gibt, könnten an sich sehr viele Formen in einer Grafik verwendet werden, jedoch ist mit üblichen Formen bereits eine Bedeutung verknüpft, was verwirrend sein kann (bspw. ein Herz als Darstellung für etwas schlechtes oder Pfeile, die auf nichts zeigen). Daher werden in der Praxis häufig „neutrale“ Formen wie Kreise, Quadrate, oder Dreiecke verwendet.

Orientierung Die *Orientierung* eines Marks kann ebenfalls genutzt werden, um Daten darzustellen (bspw. die Orientierung eines Pfeils). Natürlich kann dies nur bei bestimmten Formen genutzt werden, abhängig von den Symmetrien der Form. Außerdem ist der Übergang von einer Form mit variabler Orientierung und zwei verschiedenen ähnlichen Formen nicht strikt.

Exoten Einige Exoten unter den Channels sind bspw. Tiefe/Überdeckung, Schatten (und die Richtung dessen), Textur, Animation/Bewegung, Schattierung und Unschärfe. Diese sollen allerdings nicht oder nur sehr sparsam eingesetzt werden!

Glyphen

Glyphen sind eine spezielle Art von Marks, die insbesondere bei kartographischen Abbildungen (siehe Abschnitt 3.5) verwendet werden. Ein Glyph kann dabei beliebig komplex sein und viele Channels und Marks kombinieren, sollten dabei aber intuitiv bleiben; sie stellen eine *gemischte Visualisierung* dar. Bei Seekarten sind dies zum Beispiel die Markierungen der Tonnen auf den Seestraßen (siehe Abbildung 2.4). Formal ist ein Glyph ein „kleines, unabhängiges, visuelles Objekt, welches Merkmale von einem oder mehreren Items zeigt“. Dabei kann ein Glyph viele andere Zeichen und Visualisierungen umfassen. Da diese Definition sehr allgemein ist, muss bei dem Design von Glyphen immer zwischen Komplexität eines Glyphs und der zu erwartenden Anzahl Glyphen abgewägt werden. Dabei sollten die Glyphen einfacher sein, wenn Muster statt Werte relevant sind oder die Wertverteilung „chaotisch“ sein könnte. Wie auch bei „großen“ Visualisierungen ist es wichtig festzulegen, welche Datenattribute welche Channels nutzen (hier können alle bereits erwähnten und noch zu erwähnenden Richtlinien angewandt werden).

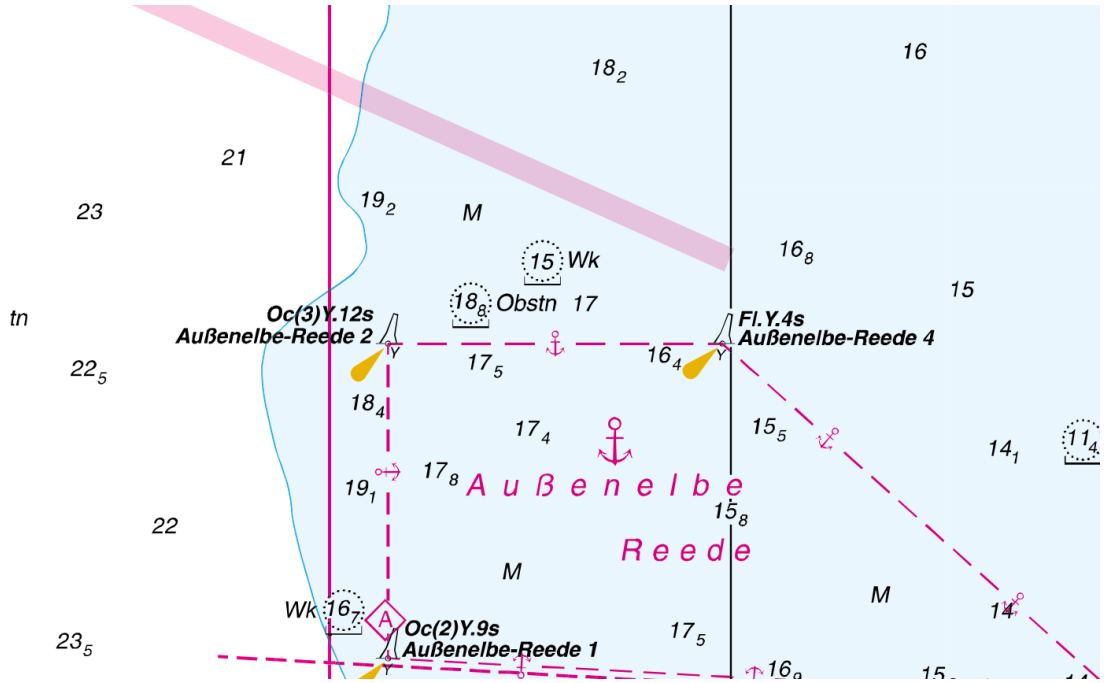

Abbildung 2.4: Auf diesem Seekartenausschnitt ist zu sehen, wie die Tonnen Außenelbe-Reede 2 und 4 durch Glyphen dargestellt werden, die viele Informationen transportieren. Bei ersterer ist bspw. die Information enthalten, dass es sich um eine gelbe einfarbige Tonne ohne Toppzeichen mit unterbrochenem Feuer in dreier-Gruppen in Gelb und einer Wiederkehr von $12\,\mathrm{s}$ handelt.

Als noch weiter gefasste Variante eines Glyphs kann ein solches sogar selbst wieder Visualisierungen enthalten, bspw. Starplots innerhalb eines Scatterplots. Häufig genutzt werden außerdem Torten-, Linien- und Balkendiagramme. Dabei nutzen die Glyphen nur den Grafikkörper, Achsen und Achsenbeschriftungen werden bewusst ausgelassen, um die Darstellung zu komprimieren.

2.2.3 Bildunterschriften

Eine *Bildunterschrift* sollte immer eine kurze Interpretation/Einordnung des Bildes mit Verweisen auf die Abbildung enthalten. Dabei sollten das Bild und die Unterschrift idealerweise auch ohne den Haupttext verständlich sein, da die Abbildungen häufig als erstes angeschaut werden. Im Haupttext sollte nachher auf die Abbildung verwiesen, diese aber nicht beschrieben werden (die Beschreibung gehört in die Bildunterschrift). Ebenfalls sollte die zentrale Aussage der Grafik im Text wiedergefunden werden.

2.3 Wahrnehmung, Position und Layout

In diesem Abschnitt werden die elementaren visuellen Aufgaben sowie die Wahrnehmung verschiedener Channels behandelt. Daraus werden abschließend Begründungen für oder gegen bestimmte Designentscheidungen abgeleitet.

2.3.1 Wahrnehmungsmodelle von Ware

Das visuelle System des Menschen ist in mehrere Schichten geteilt (V1 bis V5), die unterschiedliche Aufgaben erfüllen, wobei die zu erfüllenden Aufgaben mit steigender Schicht komplexer und abstrakter werden. Die Schichten sind dabei in beide Richtungen verbunden und die Funktion und Interaktion zwischen den Schichten ist ein offenes Forschungsthema. Diese Idee der Schichten wurde von Colin Ware zu dem *Wahrnehmungsmodell (von Ware)* zusammengefasst, welches die ablaufenden Prozesse in drei Gruppen/Stufen einteilt:

1. Parallele Verarbeitung elementarer visueller Merkmale statt. Diese Verarbeitung ist dabei nicht erlernbar und bei allen Menschen zum Großteil gleich.

2. Mustererkennung; langsame und serielle Verarbeitung, welche trainierbar ist, sodass weniger Aufmerksamkeit benötigt wird.

3. Sequentielle und zielgerichtete Verarbeitung; sehr langsame, aufmerksam gesteuerte, Übersetzung visueller Objekte in Sprache und anders herum.

Die Kommunikation zwischen den Stufen verläuft dabei ebenfalls in beide Richtungen. Da die Wahrnehmung des gleichen Bildes nicht zu jedem Zeitpunkt und jedem Menschen das gleiche Ergebnis liefert, d. h. die Verarbeitung ist nicht streng deterministisch, gibt es keine absolut „richtigen“ und „falschen“ Design für bestimmte Menschen. Eine solche Unterscheidung kann nur auf Basis gut verstandener, bei allen Menschen „gleichen“, Prozesses erfolgen, d. h. auf dem Stufe-1-System. Für eine bestimmte Menschengruppe kann eventuell auch auf Stufe 2 gewechselt werden, auf der auch bestimmte Mustererkennungen trainiert werden können.

Farbe und Farbmodelle

Ein weiterer Aspekt der Wahrnehmung ist die Farbe, die durch drei (fast) unabhängige Größen definiert wird. Je nach Farbmodell sind dies bspw. der Rot-, Grün- und Blau-Anteil (RGB-Modell) oder Farbton, Helligkeit und Sättigung (HSV[4]-Modell).

Das RGB-Modell ist dabei technisch vor allem deshalb relevant, da es einen engen Bezug zu den Rezeptoren auf der Netzhaut aufweist: Die drei Farbrezeptoren auf der Netzhaut sind S-, M- und L-Zapfen, die in der Reihenfolge kurze, mittlere und lange Wellenlängen detektieren (und damit blau, grün und grün-gelb erkennen). Andere Farben entstehen anschließend durch Mischung der Rezeptoren im Gehirn. Durch eine Kombination aller Rezeptoren wird rein die Helligkeit wahrgenommen, weshalb diese auch als unabhängiger Channel wahrgenommen wird.

Ein weiteres Farbmodell ist das CIE-Farbmodell, welche die Gesamtheit der wahrnehmbaren und darstellbaren Farben erfasst. Dabei wurde bei der Gestaltung des Modells darauf geachtet, dass „Distanzen“ im Farbraum wahrnehmungsgetreu modelliert werden. Dieses Farbmodell wird ebenfalls bei der Bewertung des Farbraums eines Monitors genutzt.

Farbskalen und Farbfehlsicht

In der Wahrnehmung und Visualisierung ist das HSV-Modell am relevantesten, da verschiedene Werte gut auf die Channels abgebildet werden können. Dabei werde innerhalb einer einzigen Visualisierung allerdings am besten immer nur ein Channel und maximal zwei verwendet. Dabei gibt es im Vergleich zu anderen Channels (abgesehen von der Position) die meisten Variationsmöglichkeiten. Dies ist gleichzeitig ein Vor- und

[4] Hue, Saturation und Value

ein Nachteil: Durch die starke Variation können sehr viele Daten abgebildet werden, allerdings sind die Skalen nicht unbedingt gut ablesbar. Durch Einschränkungen wie Farbfehlsicht werden die verfügbaren Farbskalen auf einige wenige eingegrenzt. Dabei werden drei Typen von Farbskalen unterschieden:

- *Kategorische Skala:* unterschiedliche Farbtöne; konstante Helligkeit und Sättigung
- *Divergente Skala:* zwei unterschiedliche Farbtöne für zwei Hälften der Skala (bspw. positiv und negativ); Variation von Helligkeit und Sättigung; Mitte ist hell/weiß und markiert den Nullpunkt
- *Sequentielle Skala:* konstanter Farbton, kein Gelb; monoton Helligkeits-/Sättigungsänderung

Dabei können die letzten beiden sowohl diskrete als auch kontinuierliche Daten abbilden. Bei dem Design einer Farbskala müssen verschiedene Einschränkungen wie Farbfehlsicht beachtet werden. Die üblichste Farbfehlsicht ist die Rot-Grün-Schwäche (Deuteranopie), weshalb diese Farben niemals gleichzeitig in einer Visualisierung genutzt werden sollten. Weitere Informationen zu Farbfehlsicht findet sich auf Wikipedia[5]. Um die Nutzung von Farbskalen zu vereinfachen und bekannte Effekte einzubeziehen gibt es viele Tools, bspw. ColorBrewer[6] und Adobe Color[7].

Als generelle Faustregel sollten sich die Enden der Skala immer von dem Hintergrund abheben, da diese meist am interessantesten sind. Bei einem dunklen Hintergrund sollten sie dementsprechend hell, bei einem dunklen Hintergrund (was aus unerfindlichen Gründen immer beliebter wird…) hell sein.

2.3.2 Elementare Visuelle Aufgaben

In diesem Abschnitt werden die elementaren visuellen Aufgaben behandelt, in die übergeordnete Aufgaben einsortiert werden können. Diese „high-level“ Aufgaben („Explain“, „Explore“ und „Enjoy“) repräsentieren dabei die eigentliche Ziele, die viele elementare „low-level“ Aufgaben enthalten können, die sich auf wirklich elementare Handlungen beziehen.

Aus Anwender*innensicht muss ein Mensch immer die folgenden Schritte durchgehen, um eine Aufgabe abzuarbeiten: Die Informationen in der Visualisierung suchen, die Informationen aus der Visualisierung ablesen (auch *Query* genannt) und die Informationen interpretieren (eventuell unabhängig von der Visualisierung). Dabei werden die Schritte üblicherweise nicht nur einmal, sondern mehrfach und nicht zwangsweise immer von Vorne bis Hinten durchlaufen. Dies ist in Abbildung 2.5 dargestellt. Im Idealfall unterstützt eine Visualisierung alle drei Schritte, d. h. die Aufgabe wird für den*die Nutzer*in in jedem Schritt vereinfacht[8]. Es ist jedoch möglich, sich an schlechte Visualisierungen zu gewöhnen, bzw. diese zu lernen. Daher ist eine neue – objektiv bessere – Visualisierung eventuell weniger attraktiv, da das Design noch nicht vertraut ist.

Innerhalb dieses einfachen dreischrittigen Prozesses können weiterhin die Schritte „Suchen“ und „Query“ in weitere Aufgaben eingeteilt werden, die in den nächsten beiden Abschnitten behandelt werden.

Beispiel

Suche

Die Suche kann innerhalb einer Matrix in vier Typen eingeteilt werden, siehe Tabelle 2.4: *Look-Up*, *Browsen*, *Lokalisieren* und *Explorieren*. Dabei sind alle Szenarien elementar und die Ausführung ist kurz (innerhalb einer

[5] `https://en.wikipedia.org/wiki/Color_blindness`
[6] `https://colorbrewer2.org/`
[7] `https://color.adobe.com/de/create/color-accessibility`
[8] Ein Beispiel ist die Wahl der Farbskala: Durch die Wahl einer Skala in der nur die Helligkeit/Sättigung variiert wird, wird die Frage danach, ob etwas „mehr“ oder „weniger“ ist, intuitiver als bei einer „Regenbogen“-Skala.

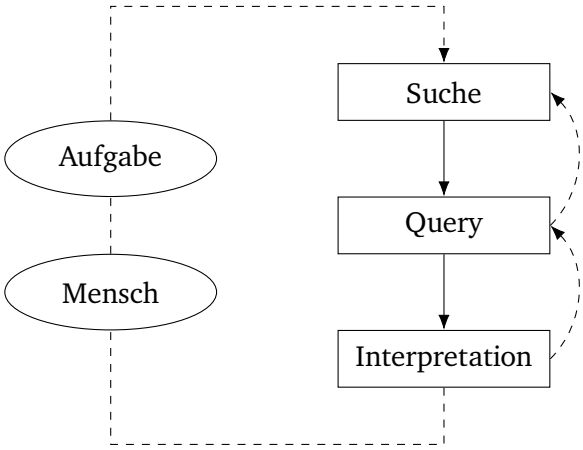

Abbildung 2.5: Aufgabenverarbeitungsprozess

	Es ist bekannt, wonach gesucht wird.	**Es sind höchstens**
Es ist bekannt, wo etwas gefunden werden kann.	*Look-Up*	
Es ist nicht bekannt, wo gesucht werden muss.	*Lokalisieren*	

Tabelle 2.4: Elementare Suchszenarien

Minute), dennoch benötigen die Aufgaben aufsteigend mehr Zeit. Dabei hängt sich die Unterscheidung nicht nur von der Aufgabe selbst, sondern auch von dem*der Anwender*in ab, d. h. von Wissen über die Inhalte der Visualisierung und der Vertrautheit mit den genutzten räumlichen Strukturen[9]. Durch die wenigen bekannten Informationen ist „Explore“ das anspruchsvollste Szenario.

Beispiel

Query

Ein Query, d. h. das Ablesen von Informationen aus einer Visualisierung, kann in sie Szenarien

- *Identifizieren:* Suche von einer Eigenschaft genau eines Items
- *Vergleichen:* Suche von Unterschieden zwischen mehreren Items
- *Zusammenfassen:* Suche von allen Items mit ähnlichen Eigenschaften

aufgeteilt werden. Dabei ist zusätzlich die Frage zu beantworten, was identifiziert/verglichen/zusammengefasst werden soll. Dies können beispielsweise einzelne Werte oder ganze „Muster“ (insbesondere in Visual Analytics) sein.

Beispiel

[9] So ist das Ablesen der Einwohnerzahl von Berlin auf einer Weltkarte für die meisten Leser*innen vermutlich eher ein Look-Up, während das Ablesen der Einwohnerzahl von Marseille eher unter „Browsen“ gefasst wird.

Einfluss der Abbildungskomponenten

Die einzelnen Komponenten einer Abbildung werden für verschiedenen Aufgaben primär verwendet: zur Interpretation werden Titel, Quelle, Legende, Achsenbeschriftungen sowie die Bildunterschrift (!) verwendet. Dahingegen fokussiert sich die Suche hauptsächlich auf den Grafikkörper, während beim Query zusätzlich die Legende sowie die Achsenwerte betrachtet werden.s

2.3.3 Eigenschaften Verschiedener Visueller Channels

In diesem Abschnitt werden die verschiedenen in Abschnitt 2.2.2 eingeführten visuellen Channels den im vorherigen Abschnitt eingeführten Such- und Queryszenarien zugeordnet bzgl. ihrer Anwendbarkeit. Die Ergebnisse davon sind in Tabelle 2.3 zusammengefasst.

Auswahl/Hervorhebung Um eine Element zu finden, also zu Lokalisieren und eventuell Identifizieren, eigen sich besonders *selektive* Channels. Dabei werden die Markierungen mit einer einzigartigen Ausprägung ohne Suchen gefunden. Ebenfalls stellt sich heraus, dass die meisten wichtigen Channels (mit Ausnahmen bei der Form) selektiv sind. Allerdings ist die Wahrnehmung stark abhängig vom Kontrast, d. h. den Eigenschaften der umliegend Items. Außerdem ist ein Channel, der zur Hervorhebung genutzt wird, nicht für andere Eigenschaften nutzbar und eine Nutzung mehrerer Channels zur Hervorhebung verschiedener Markierungen ist nicht sinnvoll.

Ordnung Um eine Ordnung zu Visualisieren, also zu Suchen und qualitativen Vergleichen, eignen sich *ordinale* Channels für ordinale Datenvariablen. Dabei müssen die Ausprägungen der Channels eine natürliche, intuitive, Ordnung aufweisen (z. B. groß/mittel/klein) und für einen Vergleich sollte keine Legende zu Rate gezogen werden müssen. Zwei Sonderfälle sind die Position, die nicht nur dem Vergleich, sondern auch der Strukturierung der Daten dient und damit jede Suche – unabhängig von dem Query – erleichtert. Außerdem kann die Orientierung sowohl für zyklische auch als linear Ordnungen genutzt werden (vgl. „Uhr“ und „Tacho“).

Differenzen Zum Vergleich von Differenzen, d. h. quantitativen Vergleichen, helfen *quantitative* Channels. Die einzigen Channels, die dies verlässlich abbilden, sind Länge und Position unter der Voraussetzung, dass beide Skalen den Nullpunkt beinhalten und linear sind. Die menschlichen Reize und deren Verhältnis zu der objektiven Reizänderung werden dabei durch *Stephens Power Law* beschrieben: Beispielsweise wird eine Sirene ($120\,\mathrm{dB}$) nicht $1\,000\,000$-mal lauter wahrgenommen als ein Gespräch ($60\,\mathrm{dB}$), obwohl sie entsprechend viel energiereicher ist. Dies wird durch den Exponent α charakterisiert, welcher ausschließlich für die Länge $\alpha = 1$ ist.

Zusammenfassen Zur Zusammenfassung von Daten, also zum Browsing/Exploring und Zusammenfassen, eigen sich *assoziative* Channels sehr gut, da sie helfen, ähnliche Items als Gruppe wahrzunehmen. Bei „Explain“-Aufgaben kann die Ähnlichkeit dabei durch eine Abbildung auf kategorische Variablen vorgegeben werden, bei „Explore“/„Enjoy“ kann die Ähnlichkeit auch eine kombinierte Wahrnehmung von Mustern in verschiedenen Channels sein. Dabei kann die Wahrnehmung verschiedener Muster allerdings kollidieren, was bei optischen Täuschungen ausgenutzt wird.

2.3.4 (Ungewollte) Einflüsse

In diesem Abschnitt werden einige (oft ungewollte) gegenseitige Einflüsse von verschiedenen Channels aufeinander behandelt. Diese räumlich beieinander liegen, damit die Wahrnehmung weniger beeinflusst wird. Dabei ist die Wahrnehmung fast immer von dem Verhältnis von Kontrast und Entfernung abhängig – und vom Kontext.

Kontrast, Perzeptuelle Länge und Farbnamen Die *perzeptuelle Länge* beschreibt dabei die Anzahl unterscheidbarer Werte je Channel, auch wenn die Unterscheidung erschwert wird. Üblicherweise sind das nicht viele, was insbesondere bei der Visualisierung verschiedener Kategorien relevant ist. Im Allgemeinen sind solche Farben und Formen unterscheidbar, die man benennen kann. Es sollten aber eigentlich nicht die Farben, sondern die Kategorien unterschieden werden. Um dies nicht zu erschweren, sollten so wenig Farben und Formen wie möglich verwendet werden.

Gesichter, Bewegung und 3D aus Bildern Da Menschen sehr gut darin sind/sein müssen, Gesichter schnell zu erkennen, neigen sie dazu, diese überall zu erkennen, was Darstellungsschwierigkeiten hervorrufen kann. Die zweit-wichtigste natürliche Aufgabe des Sehsinns ist die Erkennung von Bewegung und 3D-Grafiken auch aus „flachen“ Bildern. Dies lässt sich nicht abschalten und beeinflusst die Wahrnehmung aller Channels.

Separierende und Integrierende Channels

Werden verschiedene visuelle Channels gleichzeitig eingesetzt, so ist zwischen *separierende* und *integrierenden* Channels zu unterscheiden: Bleibt von den einzelnen Variablen etwas übrig (*separierend*) oder entsteht ein neuer Sinneseindruck (*integrierend*)? Beispielsweise ist Position und Form ein separierendes Paar, wohingegen der Rot- und Blau-Kanal einer Farbe integrierend sind. Je nach Anwendung ist beides sinnvoll, da separierende Paare mehr Datenvariablen, integrierende Paare hingegen mehr unterscheidbare Werte, erlauben. Dabei ist, wie so oft, der Übergang zwischen den Kategorien fließend und abhängig vom konkreten Anwendungsfall. Eine kleine Übersicht ist in Abbildung 2.6 gegeben.

2.3.5 Position, Layout und Komposition: Suchen vs. Finden

Die zentrale Idee von geschicktem Layouting ist, die Suche zu erleichtern oder zu umgehen (von „Suchen“ zu „Finden“). Ein Beispiel ist die Anordnung von einzelnen Diagrammen so zu gestalten, dass ähnliche räumliche Strukturen nebeneinander liegen (z. B. gemeinsame y-Achsen). Dadurch kann eine einmal „gelesene“ Struktur auf die benachbarten Visualisierungen angewendet und zeitsparend genutzt werden.

Beispiele

Zusammengesetzte Visualisierungen

Zusammengesetzte Visualisierungen können genutzt werden, um komplexe Daten abzubilden. Dabei gibt es verschiedene Varianten bis hin zu interagierenden Visualisierungen (siehe Abbildung 2.7). Dabei gibt es einige visuelle Designmuster, z. B.:

- *Juxtaposition (Gegenüberstellung):* häufigstes Designmuster; die View stehen nebeneinander, die Verbindung ist implizit (z. B. durch eine gemeinsame Achse)

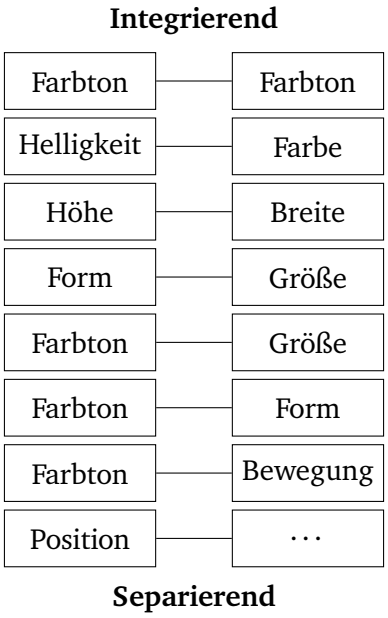

Abbildung 2.6: Vergleich separierender und integrierender Channels

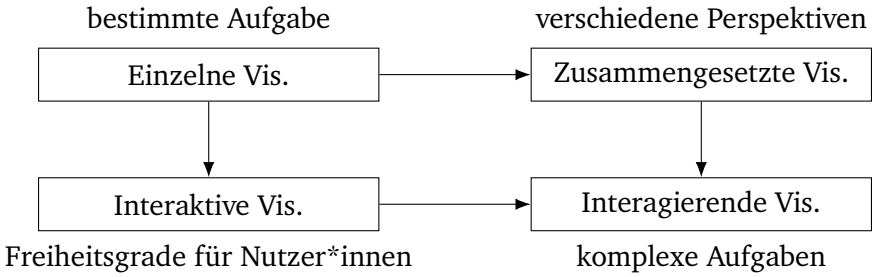

Abbildung 2.7: Verschiedene Arten von zusammengesetzten Visualisierungen

- *Superimposition (Überlagerung):* zwei Views nutzen den gleichen Raum, dadurch wird ein räumlicher Bezug hergestellt; gemeinsame Achse oder Markierungen
- *Overloading (Überladung):* eine Visualisierung wird ergänzend in der Hauptvisualisierung dargestellt; Nutzung der gleichen visuellen Abbildung auf die Position, aber ggf. andere Marks/Channels
- *Nesting (Einbettung):* Marks sind selbst (kleine) Visualisierungen; stellt einen Bezug zwischen „Überblick“ und „Detail“ her

Beispiele

2.4 Interaktion

In diesem Abschnitt werden verschiedene Methoden beschrieben, um Interaktion mit einer Visualisierung zu ermöglichen. Diese ist hilfreich, da unmöglich auf alle Fragestellungen, die Nutzer*innen haben könnten,

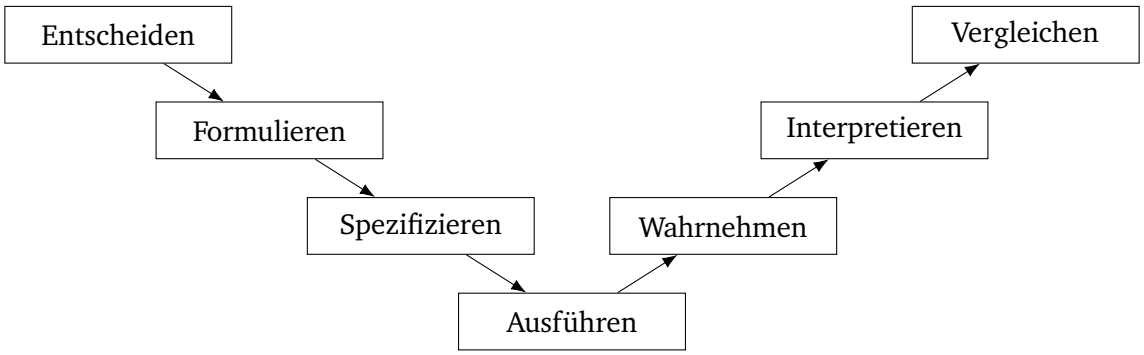

Abbildung 2.8: Interaktionsmodell nach Norman

einzugehen bei dem Design der Visualisierung. Außerdem ist es häufig nicht möglich, alle Daten auf einmal zu zeigen, die durch Interaktion eingeblendet werden können. Ebenfalls kann Interaktion der Exploration dienlich sein. Im Informationsvisualisierungsprozess (Abbildung 2.1) wird *Interaktion* durch den rückführenden Pfeil von Mensch zu den einzelnen Schritten eingebunden. Dabei ist wichtig zu unterscheiden, dass dies nicht die „Interaktionsmöglichkeiten“ eines*einer Entwicklers*Entwicklerin beschreibt, sondern die direkte Interaktion durch eine*n Nutzer*in.

2.4.1 Benutzungsschnittstellen

Der Begriff des *User-Centered Design* wurde zuerst von Donald Norman geprägt, der sich mit dem Design von Gegenständen, Aufmerksamkeit und vielen weiteren Themen beschäftigt hat. Dabei hat er ein Modell zur Interaktion entwickelt, welches in Abbildung 2.8 dargestellt ist. Demnach durchläuft ein Mensch die folgenden Schritte bei der Ausführung einer Interaktion:

1. Entscheiden, was zu tun ist
2. Formulieren einer Absicht
3. Spezifikation einer Handlung
4. Ausführen einer Handlung
5. Wahrnehmung der Reaktion des Systems
6. Interpretation des Systemzustands
7. Vergleich zwischen dem tatsächlichen Zustand und dem ursprünglichen Ziel

Dabei gibt es viele Probleme bei den Übergängen zwischen den Stufen. Zunächst sollte klar sein, welche Funktionen und Interaktionen möglich sind und wie diese durchgeführt werden können (Icons, Buttons, etc.). Des Weiteren sollte der Zustand des Systems leicht erkennbar sein. Solche *Benutzer*innenschnittstellen* sind nach DIN EN ISO 9241-110 alle Bestandteile eines interaktiven Systems, die Informationen und Steuerelemente zur Verfügung stellen, die notwendig sind, um eine bestimmte Aufgabe zu erledigen.

Interaktionsmodi nach Spence

Robert Spence hat mehrere Modi der Interaktion (*Interaktionsmodi*) identifiziert, mit denen ein System modifiziert werden kann. Diese sind *kontinuierliche*, *schrittweise*, *passive* und *gemischte* Interaktionen.

Kontinuierliche Interaktion Bei der kontinuierlichen Interaktion ändert sich die Visualisierung kontinuierlich und direkt (bspw. ein Zoom). Dafür ist es wichtig, dass das System hochperformant reagiert.

Schrittweise Interaktion In der schrittweisen Integration findet die Interaktion entlang verschiedener Schritte statt (z. B. in der Navigation: Was sind mögliche Schritte von hier aus?). Dabei definiert Spence zwei Arten von *Sensitivity*, die, motiviert durch die Sozialwissenschaft in der „Sensitivity“ die zwischenmenschliche Aufmerksamkeit beschreibt, die Aufmerksamkeit beschreibt, welche Bewegungen ausführbar sind (Movement, SM) und welche Handlungen dafür nötig sind (Interaction, SI). Eine wichtige Komponente ist damit die *Affordance*, also der „Aufforderungscharakter“ von Objekt sowie *Residues*, Hinweise auf entfernte Funktionen. Beim Interaktionsdesign müssen diese Faktoren bestmöglich unterstützt werden, um eine angenehme Interaktion zu gestalten.

Passive Interaktion Passive Interaktion baut auf dem Aspekt auf, dass Nutzer*innen viel Zeit mit Augenbewegungen, Wahrnehmung und Verarbeitungsprozessen verbringen. Daher beinhaltet eine passive Interaktion auch sich ändernde Repräsentationen, aus denen die Nutzer*innen hilfreiche Informationen ziehen. Beispiele sind statistische Darstellungen, Browsing und nicht direkt von dem*der Nutzer*in beeinflusste Bewegungen.

Gemischte Interaktion Gemischte Interaktionen können, wie der Name schon sagt, sämtliche anderen Interaktionsmodi enthalten und kombinieren.

Akzeptable Antwortzeiten Da das System eine gewisse Zeit benötigt, um auf eine Anfrage oder Interaktion zu reagieren, wurde untersucht, was die noch akzeptablen Antwortzeiten sind. Dabei wurden die folgenden Werte gefunden:

- Animationen, fließende Bewegungen und kontinuierliche Interaktion: $0.1\,\mathrm{s}$
- Reaktion auf Benutzer*innenaktionen: $1\,\mathrm{s}$
- Komplexe Anfragen: $5\,\mathrm{s}$ bis $30\,\mathrm{s}$

Für sämtliche Aktionen die länger als $1\,\mathrm{s}$ dauern, sollte dabei ein Fortschrittsbalken hinzugefügt werden.

2.4.2 Interaktionstechniken

Nachdem im vorherigen Abschnitt die grundlegenden theoretischen Ergebnis im Bezug auf Interaktion behandelt wurden, werden in diesem Abschnitt einige konkrete Techniken behandelt. Dabei gibt es viel Gestaltungsspielraum und verschiedene Taxonomien, z. B.:

- Systemnahe Interaktionstechniken
- Dimensionen der Interaktionstechniken (Spence; Interaktionsmodi)
- Interaktionsoperatoren (Ward und Yang; Interaktionsräume: screen-space, data-value-space, attribute-space, etc.)
- Benutzer*innenaufgaben (Amar, Eagon, Stasko; Werte abrufen, Filtern, Extrema finden, Anomalien finden, Korrelieren, etc.)
- Kategorien der Interaktion (Yi et al.)

In folgenden Abschnitten werden *systemnahen* Interaktionstechniken und *Kategorien* von Interaktion behandelt.

Systemnahe Interaktionstechniken

Die systemnahen Interaktionstechniken teilen sich weiter auf in Selektion, Navigation, Shneidermans Mantra, Fokus und Kontext, Überblick und Detail sowie Brushing und Linking:

Selektion Das Ziel der Selektion ist Identifikation eines bestimmten Objekts oder Definition einer (beliebigen) Teilmenge. Dafür gibt das System Feedback in Form einer Markierung mit einer geeigneten visuellen Auszeichnung (z. B. einkreisen). Das *Fitts Law* ist dabei die Zeit, die zum Selektieren benötigt wird,

$$Selektionszeit = a + b \log_2\left(\frac{D}{W} + 1\right),$$

wobei D die Distanz zum Zentrum und W die Größe/Ausdehnung des Ziels sind und a und b empirisch ermittelte Konstanten.

Navigation Zur Navigation gibt es viele verschiedene Optionen, bspw. Tastatur, Maus/Touchscreen (2D-absolut), Space-Maus (6D-relativ), Trackingsysteme (6D-relativ/absolut) sowie Kombinationen dieser (z. B. werden Computerspiele üblicherweise mit Tastatur und Maus gesteuert). Dabei gibt es wiederum verschiedenste Kamerametaphern wie World-in-Hand, Eyeball-in-Hand, „Gehende“ Kamera oder „Fliegende“ Kamera. Alternative können Zoom-und-Pan-Methoden wie in Google Maps oder verschiedenste andere Darstellungsformen verwendet werden. Hilfreich dabei sind Visualisierungen als Orientierungshilfe (lokal und global).

Werden Zoom-Verfahren (die auch der Navigation angehören) verwendet, wird zwischen geometrischem Zoom (der „nur“ Objekte vergrößert) und semantischem Zoom (wodurch mehr Daten angezeigt werden) unterschieden.

Shneidermans Mantra

1. Überblick über alle Daten
2. Zoom und Filter
3. Details auf Anfrage

Demnach sollen zunächst alle Daten in grober Art und Weise angezeigt werden, aus denen der*die Nutzer*in anschließend relevante Daten auswählen (Filtern) und sich die Details dieser anzeigen lassen kann. Eine alternative Version für die visuelle Suche ist „Suche, Zeige Kontext, Expandiere auf Anfrage“.

Fokus und Kontext Fokus und Kontext ist eine Art der Interaktion die Raum schafft abhängig von Benutzer*innenfokus. Dies wird oft realisiert durch eine „Fischauge“-Darstellung, bei der lokal der Detailgrad erhöht wird. Diese Technik ist gut kombinierbar mit Level of Detail-Methoden und fügt sich gut in Shneidermans Mantra ein. Durch die Fischaugen- oder Magic Lens-Darstellung wird der Kontext beibehalten, was bei dem Verständnis hilft. Die Magic Lens-Darstellung ist dabei ähnlich zu einer Lupe, d. h. es werden lokal Details hervorgehoben, aber die Umgebung wird – im Gegensatz zum Fischauge – nicht verzerrt. Allerdings wird beiden Fällen die Wahrnehmung räumlicher Eigenschaft beeinflusst.

Überblick und Detail Bei dieser Interaktionstechnik werden mehrere Visualisierungen hierarchisch gekoppelt, wobei lokal die Detailstufe erhöht wird. Dabei bleiben räumliche Beziehungen erhalten indem die „höhere“ Visualisierung klein eingeblendet wird. Dies hat den Nachteil, dass eventuell Details verdeckt werden.

Brushing und Linking Bei Brushing und Linking werden bei der durch Auswahl (Brushing) von Datenpunkten in einer Visualisierung die korrespondierenden Daten in einer anderen Visualisierung hervorgehoben (Linking). Diese Technik ist daher insbesondere für Visualisierungen hilfreich, in denen mehrere Perspektiven der gleichen Daten angezeigt werden.

Kategorien der Interaktion

Nach Yi et al. sind die Kategorien der Interaktion wie folgt, wobei das Ziel dieser Kategorisierung eine stärkere Verbindung von Interaktionstechnik und Aufgaben ist:

- Selektion: „markiere etwas interessantes“
- Exploration: „zeige etwas anderes“; Exploration
- Rekonfiguration: „zeige eine andere Zusammenstellung“
- Enkodierung: „zeige eine andere Repräsentation“
- Abstrahieren/Spezialisieren: „zeige weniger oder mehr Details“; Überblick und Detail
- Filtern: „zeige etwas unter Bedingungen“; Shneidermans Mantra
- Bezug: „zeige Beziehungen der Elemente“; Brushing und Linking

2.4.3 Design

In diesem Abschnitt wird das tatsächlich *Design* der Interaktion behandelt, also die Umsetzung als User Interface.

Leitsätze

Die Leitsätze der Human-Computer-Interaction sind aufgeteilt in die *Navigation*, *Organisation der Anzeige*, *Erzeugung von Aufmerksamkeit* sowie der *Unterstützung der Dateneingabe*:

Navigation Im Nutzungskontext „Web“ haben sich die folgenden „Best Practices“ im HCI etabliert:

- Arbeitsabläufe sollten stets standardisiert sein.
- Bei eingebetteten Links sollte eine klare Beschreibung des Ziels vorliegen.
- Überschriften sollten sprechend und eindeutig sein.
- Für eine ausschließende Auswahl sollten Radio Buttons verwendet werden.
- Websites sollten ohne weiteres druckbar sein.
- Bei größeren Bildern sollten Thumbnails als Vorschau gezeigt werden.

Organisation der Anzeige

- Die Anzeige sollte Konsistent sein.
- Die Informationsaufnahme durch den*die Benutzer*in sollte möglichst effizient sein.
- Das Gedächtnis sollte nur minimal belastet werden, d. h. Informationen sollten immer da vorhanden sein, wo sie benötigt werden.
- Anzeigen sollten flexibel und individualisierbar sein.
- Wo immer möglich sollte eine grafische Darstellung anstelle von Texten und Zahlen verwendet werden.
- Das Design sollte schwarz-weiß gehalten werden, wobei Farben genau dann eingeführt werden, wenn diese hilfreich sin.
- Der*die Nutzer*in muss beim Design involviert werden!

Erzeugung von Aufmerksamkeit Die Erzeugung von Aufmerksamkeit, um diese des*der Nutzers*Nutzerin zu lenken stellt einen wichtigen Faktor dar, damit der*die Nutzer*in auf der Website nicht „verloren“ geht. Die folgenden Leitlinien haben sich diesbezüglich etabliert:

- Intensive Farben sollten für wichtige Aspekte reserviert werden.
- Markierungen
- Größe
- max. drei Schriftarten
- Blinkende Elemente (mit $2\,\mathrm{Hz}$ bis $4\,\mathrm{Hz}$) nur in limitierten Bereichen
- max. vier Standardfarben, alles weitere nur für die gelegentliche Nutzung
- Weiche Töne für positives, harte Töne für negatives Feedback

Generell gilt bei dem erhaschen von Aufmerksamkeit: Weniger ist mehr!

Unterstützung der Dateneingabe

- Konsistenz der Dateneingaben
- Minimierung der Nutzer*innenaktionen
  - Mausklick statt Befehlseingabe
  - Auswahl aus Listen statt Freitext
  - Möglichst wenige Wechsel des Eingabegerätes (Expert*innen nutzen häufig lieber die Tastatur als zur Maus zu greifen)
- Möglichst wenig Nutzung des Gedächtnisses
- Die Dateneingabe sollte flexibel und individualisierbar sein.

Prinzipien

Neben den obigen Leitlinien zum Interaktionsdesign haben sich noch eine Prinzipien entwickelt, wie das Design angegangen werden sollte:

Ermittlung des fachlichen Niveaus des*der Nutzers*Nutzerin Die Ermittlung des fachlichen Niveaus ist essentiell zur effizienten Gestaltung des User Interface:

- Anfänger*innen und Erstnutzer*innen benötigen intensive Hilfsdialoge, die die Abläufe erklären.
- Sachkundige, gelegentliche Nutzer*innen benötigen konsistente und wiederkehrende Abläufe sowie sinnvolle, aber knappe Meldungen.
- Expert*innen benötigen schnelle Antwortzeiten, Shortcuts, Abkürzungen, Feedback im Hintergrund statt Vordergrund und andere beschleunigende Dialoge.

Ermittlung der Arbeitsaufgaben Bei der Ermittlung der Aufgaben ist die Häufigkeit dieser ein wichtiger Maßstab bei dem Interface-Design. Häufige Aufgaben sollten dabei schnell erledigt werden können (z. B. Kopieren und Einfügen), während weniger häufige oder seltene Aufgaben auch länger dauern dürfen.

Wahl des Interaktionsstils Bei der Wahl des Interaktionsstils gibt es einige Optionen, z. B. Kommandozeile, Eingabeformular, Menüauswahl, Direkte Manipulation und Spracherkennung.

Die Acht Goldenden Regeln der Gestaltung

1. Konsistenz wo immer möglich (Abläufe, Farben, Layout, Fonts, Menüs, etc.)
2. Universelle Benutzbarkeit (Anfänger*innen bis Expert*innen, Altersgruppen, Behinderungen, technische Vorlieben, etc.)
3. Informatives Feedback für jede Aktion
4. Design von Dialogen, die mehrere Aktionen zum Abschluss führen (Beschleunigung)
5. Verhinderung von Fehlern (z. B. invaliden Konfigurationen)
6. Einfaches rückgängig machen von Aktionen
7. Unterstützung eines Gefühls der Kontrolle über die Schnittstelle
8. Reduktion der Gedächtnisbelastung (Daumenregel: ca. sieben Informationen können zeitgleich im Arbeitsgedächtnis gehalten werden)
    - Nicht zu viele Informationen auf einer Anzeige
    - Rasche Wechsel führen zu Change Blindness, d. h. andere, potentiell wichtige, Änderungen werden übersehen

Menschliche Reaktionszeit

Empirisch wurde das *Nick-Hyman-Gesetz* zur menschlichen Reaktionszeit ermittelt. Diese beträgt demnach

$$Reaktionszeit = a + b \log_2(C),$$

wobei C die Anzahl Auswahlmöglichkeiten und a und b empirisch zu ermittelnde Konstanten sind. Dürfen Menschen gelegentlich Fehler machen (z. B. wenn Aktionen zurückgenommen werden können), so kann die Reaktionszeit deutlich erhöht werden.

3 Spezialisierte Visualisierungstechniken

In diesem Kapitel werden einige für bestimmte Daten spezialisierte Visualisierungstechniken behandelt, bspw. für hochdimensionale oder sehr viele Daten.

3.1 Hochdimensionale Daten

Zu Beginn muss die Frage gestellt werden, was „hochdimensional“ ist. Denn: „Hoch“dimensional kann sogar schon drei quantitative Attribute bedeuten: Ein Scatterplot kann bspw. problemlos zwei quantitative Dimensionen Dimension darstellen, bei drei quantitativen – und spätestens bei vieren – wird es jedoch schwer. Selbst wenn die dritte Dimension nominal ist, wird diese unbewusst als abhängige Variable wahrgenommen, was ggf. nicht gewollt ist. Eine Möglichkeit ist die Darstellung in einem 3D-Plot, aber erstens bricht dies spätestens bei vier Dimensionen zusammen und zweitens sind diese Art von Visualisierungen schwer zu lesen – auch wenn dies durch Hilfslinien oder Interaktion (bspw. Rotation) verbessert werden kann.

Grundsätzlich gilt: Fast alle Techniken zur Darstellung hochdimensionaler Daten sind nicht dazu geeignet, einzelne Datenwerte abzulesen sondern dienen in erster Linie dazu Trends und Verteilungen sichtbar zu machen, um Muster, Ausreißer und/oder Abhängigkeiten zu finden. Sollen zusätzlich Werte abgelesen werden, ist dies durch Interaktion (z. B. Tooltips) realisierbar.

Eine Übersicht über die vorgestellten Techniken ist in Tabelle 3.1 zu finden.

3.1.1 Quantitative Daten

Dieser Abschnitt behandelt Visualisierungstechniken für hochdimensionale quantitative Daten, d. h. Daten mit vielen quantitativen Attributen. Im Allgemeinen lassen sich diese Verfahren jedoch alle auf kategorische Daten anwenden.

Scatterplot-Matrix

Eine der bekanntesten Techniken ist die *Scatterplot-Matrix*, in der alle Attribute sowohl auf die x- und y-Position als auch die Attributsnamen selbst auf diese abgebildet werden. Dadurch ergibt sich eine Matrix an Scatterplots, die für sich genommen jeweils ein Attributspaar darstellen (siehe Abbildung 3.1).

Stärken

- geeignet für 4 bis 8 Dimensionen
- getreue Darstellung paarweiser Abhängigkeiten
- komplexe Abhängigkeiten sind über Brushing und Linking darstellbar
- weitgehend unabhängig von der Zeilen/Spalten-Anordnung
- kaum Verzerrung der Wahrnehmung

Name	Dimensionstyp	#Dimensionen	#Datensätze	Anord.?	Lesb. (Int.)	Aufgabe
Scatterplot-Matrix	quantitativ	4 bis 8	100 bis 10 000	++	+ (++)	paarweise Korrelationen finden
Starplot	quantitativ[2]	4 bis 15	10 bis 100	− − −	○ (+)	Vergleich in vielen Dimensionen
Parallele Koordinaten	quantitativ[2]	ca. 30[1]	100 bis 1000	−−	− (+/++)	paarweise Korrelationen finden Suchen in nD
RadViz	quantitativ	4 bis 15	10 bis 1000	− − −	− (○/−)	lineare Abhängigkeiten finden
Parallel Sets	nominal[3]	ca. 30[1]	„unendlich“	−−	○ (+/++)	paarweise Korrelationen finden Suche in nD
Mosaik-Plot	nominal[3]	4 bis 6	„unendlich“	−−	− (○)	Häufigkeiten vergleichen paarweise Abhängigkeiten finden
KV-Map	nominal[3]	4 bis 10	„unendlich“	−	−− (○)	Korrelationen in nD finden Muster in nD finden
Tabellen	alle	30+	100 bis 1000	+	+ (++)	Vergleichen, Korrelationen finden

Tabelle 3.1: Übersicht über Visualisierungstechniken für hochdimensionale Daten
[1] limitiert durch die Bildbreite/-höhe
[2] auch nominale Daten können mit einer beliebigen Anordnung verwendet werden
[3] durch Diskretisierung können auch quantitative Daten verwendet werden

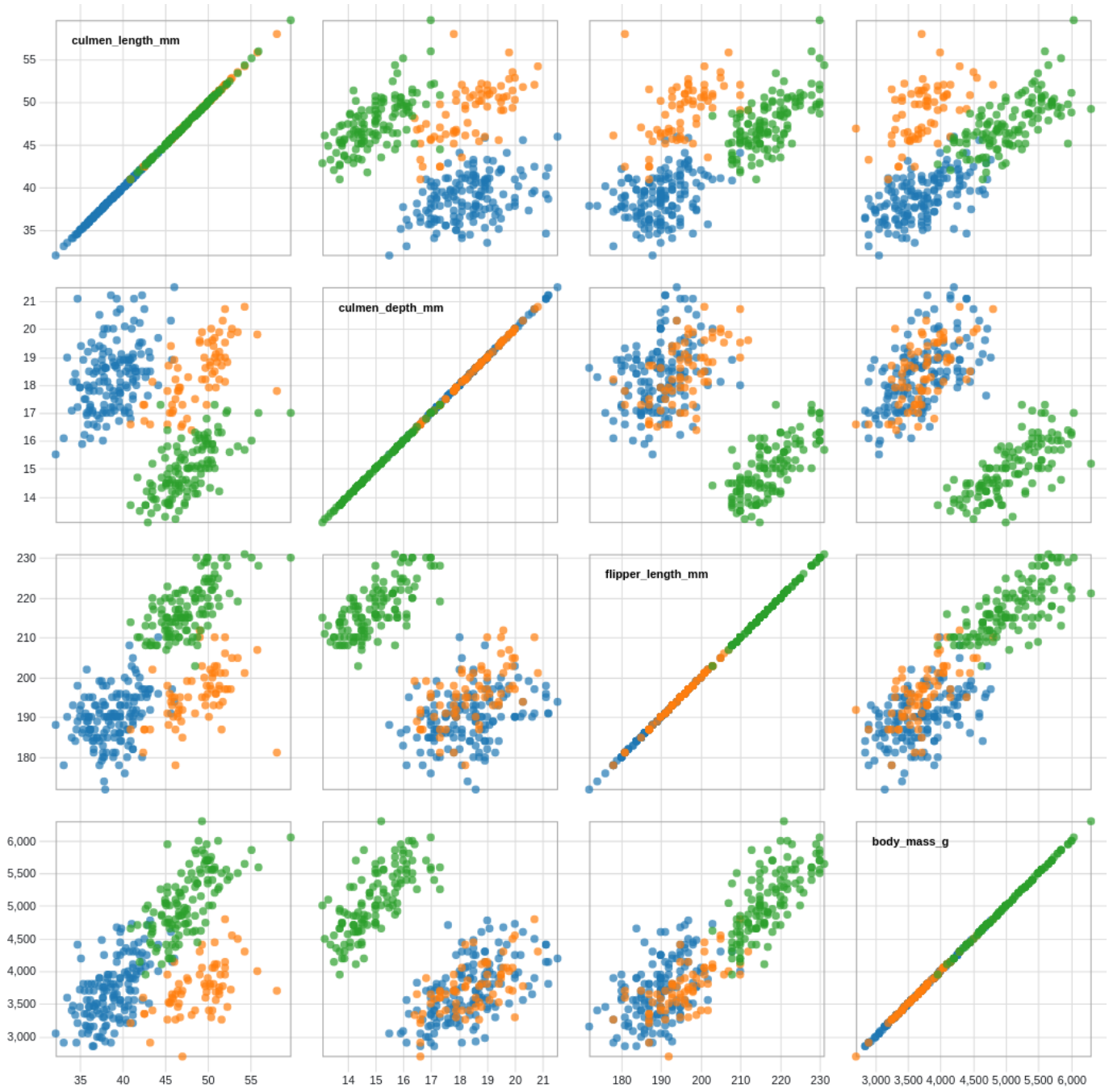

Abbildung 3.1: Visualisierungstechnik (nD-Quantitativ): Scatterplot-Matrix
Quelle: `https://observablehq.com/@d3/brushable-scatterplot-matrix`

Schwächen

- skaliert schlecht für viele Datenpunkte (Overplotting)
- zeigt „nur“ paarweise Abhängigkeiten

Starplot

Die Idee des *Starplots* ist, unabhängige Dimensionen radial um einen gemeinsamen Ursprung anzuordnen und die Daten als Linie zwischen diesen Achsen darzustellen (siehe Abbildung 3.2). Um mehrere Daten zu visualisieren können entweder mehrere Farben verwendet werden oder mehrere kleine Plots, sogenannte *Small Multiples*[1] erstellt werden. Dabei wird allerdings die Vergleichbarkeit schwerer, die jedoch durch eine geeignete Anordnung der Plots (ähnliche nebeneinander) zu teilen wiederhergestellt werden kann. Eine weitere Technik zur Verbesserung der Vergleichbarkeit ist das Malen einer Fläche statt einer Linie, da dies die Formwahrnehmung verstärkt.

Stärken

- geeignet für 4 bis 15 Dimensionen
- Ähnlichkeit über Formwahrnehmung

Neutral

- einfache Korrelationen zwischen benachbarten Achsen sind gut sichtbar

Schwächen

- skaliert schlecht für viele Datenpunkte
- Abhängig von der Anordnung der Achsen
- nur Korrelation benachbarter Achsen sichtbar

Parallele Koordinaten

Parallele Koordinaten folgen der gleichen Idee wie Starplots, „rollen“ die Achsen jedoch ab, d. h. sie werden nebeneinander dargestellt (siehe Abbildung 3.3). Diese Technik ist sehr populär und wird – im Vergleich zu Starplots – eher selten in „Small Multiples“ eingesetzt. Das größte Problem bei parallelen Koordinaten ist dabei das Overplotting, d. h. der Plot wird nicht lesbar, da zu viele Daten angezeigt werden. Dieses Problem kann allerdings sehr gut durch Interaktionen (z. B. Brushing und Linking) abgeschwächt werden, indem bspw. eine durchgehende Linie hervorgehoben wird. Auch helfen Filter und interaktiv sortierbar Achsen weiter.

Stärken

- im Prinzip beliebig viele Achsen
- einfache Korrelationen zwischen benachbarten Achsen sind gut sichtbar

[1] Small Multiples ist eine generelle Art, viele Daten darzustellen und beschränkt sich nicht nur auf Starplots.

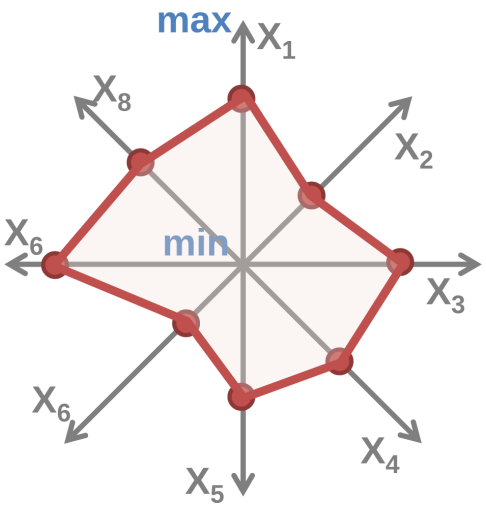

Abbildung 3.2: Visualisierungstechnik (nD-Quantitativ): Starplot
Quelle: Vorlesungsfolien

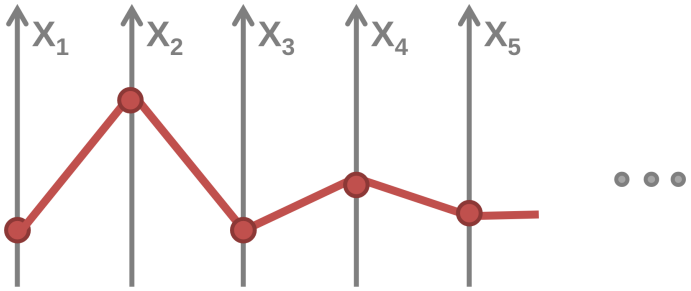

Abbildung 3.3: Visualisierungstechnik (nD-Quantitativ): Parallele Koordinaten
Quelle: Vorlesungsfolien

Schwächen

- skaliert schlecht für viele Datenpunkte
- Linienzüge können in Kreuzungen nicht verfolgt werden
- Abhängig von der Anordnung der Achsen
- Achsenrichtung beeinflusst die Wahrnehmung
- nur Korrelation benachbarter Achsen sichtbar

RadViz

RadViz nutzt, ebenso wie ein Starplot, ein radiales Layout der Achsen, allerdings werden keine Linien verbunden, sondern hohe Werte in einer Achse „ziehen“ den Punkt in Richtung dieses Achsenendes, sodass ein Datenpunkt am Ende nur als ein Punkt angezeigt wird (siehe Abbildung 3.4). Zur Erstellung eines solchen Plots werden die Daten zunächst min-max-normiert und anschließend die Achsenvektoren entsprechend

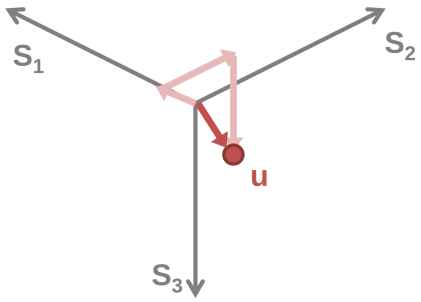

Abbildung 3.4: Visualisierungstechnik (nD-Quantitativ): RadViz
Quelle: Vorlesungsfolien

gewichtet, d. h.

$$u_1 = \frac{\sum_{j=1}^{n}}{y_j \cos(\alpha_j)} \sum_{j=1}^{n} y_j \qquad u_2 = \frac{\sum_{j=1}^{n}}{y_j \sin(\alpha_j)} \sum_{j=1}^{n} y_j$$

wobei α_j den Winkel der j-ten Achse und u_1 und u_2 die horizontale bzw. vertikale Position darstellen.

Stärken

- platzsparend
- relative Datenverhältnisse sind gut sichbar

Schwächen

- Achsenanordnung bestimmt die Position
- redundante Achsen verzerren das Ergebnis
- Punktposition ist nicht eindeutig aufgrund der Normierung

3.1.2 Kategorische Daten

Dieser Abschnitt behandelt Visualisierungstechniken für hochdimensionale kategorische Daten, d. h. Daten mit vielen nominalen oder ordinalen Attributen. Im Allgemeinen lassen sich dafür ebenfalls alle Verfahren aus dem vorherigen Abschnitt anwenden, es gibt jedoch auch einige spezielle Verfahren.

Parallel Sets

Parallel Sets sind ähnlich zu parallelen Koordinaten, stellen jedoch Datenmengen mit einem bestimmten Attributswert dar, die sich an jeder Achse aufteilen. Des weiteren werden die Achsen oft übereinander statt nebeneinander angeordnet (siehe Abbildung 3.5). Durch die Einfärbung nach dem ersten Attribut kann die Zugehörigkeit auch später noch zuordnen. Durch Transparenz wird zusätzlich sichergestellt, dass der Verlauf auch bei Überschneidungen noch sichtbar ist.

Durch Interaktion lassen sich, wie bei parallelen Koordinaten, einige der unten aufgelisteten Nachteile entschärfen. Dabei kann bspw. die Achsen- und Kategorienreihenfolge verändert sowie Teilmengen hervorgehoben werden. Außerdem könnte die Kategorie, welche die Farbe bestimmt, auswählbar gemacht werden.

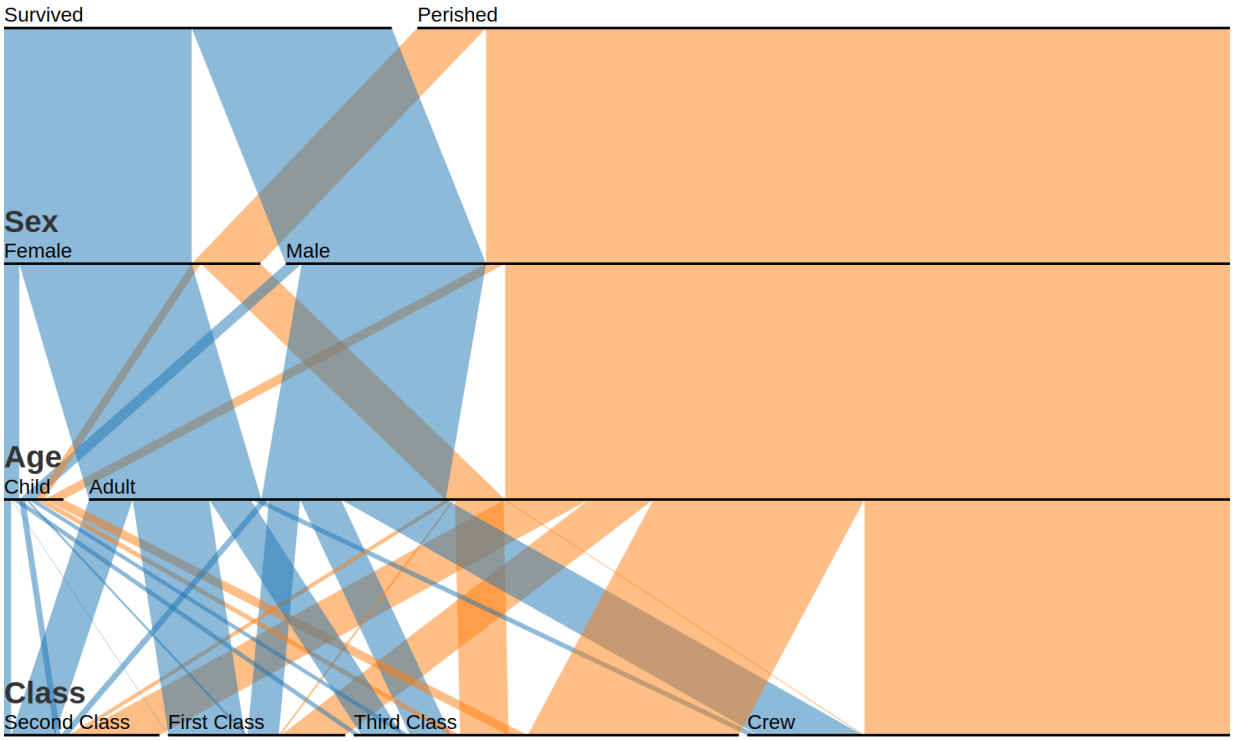

Abbildung 3.5: Visualisierungstechnik (nD-Kategorisch): Parallel Sets
Quelle: `https://www.jasondavies.com/parallel-sets/`

Stärken

- skaliert für beliebig viele Daten (da die Anzahl angezeigter Linien nur von der Anzahl Kategorien und Werte in diesen abhängt)
- geeignet für 20 und mehr Dimensionen

Schwächen

- Abhängig von der Anordnung der Achsen
- Abhängig von der Anordnung der Kategorien
- häufige Kategorien dominieren seltene (Overplotting)
- Abhängigkeit von der Zeichnungsreihenfolge (durch Überlagerung; Overplotting)

Mosaik-Plot

Ein *Mosaik-Plot* basiert auf der rekursiven Teilung von Flächen, d. h. für die Werte eines Attributs wird die Wertmenge horizontal oder vertikal geteilt, was eine neue Untermenge ergibt. Dabei wird bei jeder Teilung in anderes Attribut gewählt, bis alle darzustellenden Attribute aufgebraucht sind. Der Ort der Teilung entspricht dabei, wie bei den Parallel Sets, der Häufigkeit (siehe Abbildung 3.6).

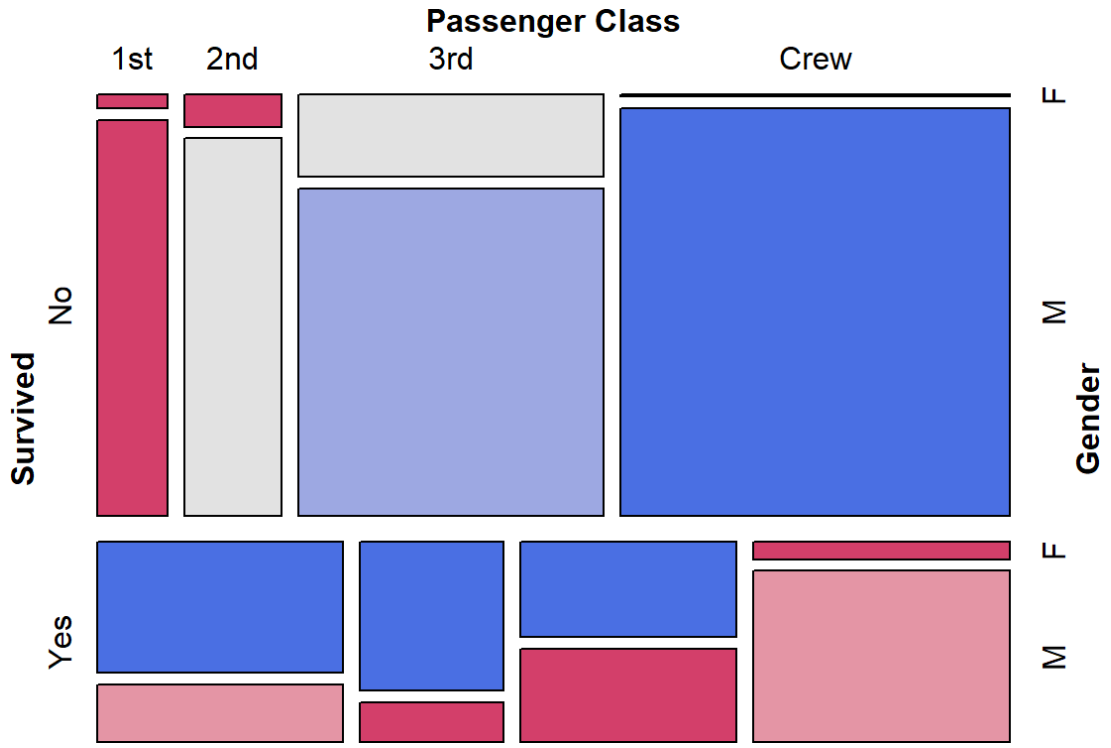

Abbildung 3.6: Visualisierungstechnik (nD-Kategorisch): Mosaik-Plot
Hinweis: Die Farbe innerhalb der Kacheln kann weitere Daten darstellen; die Skala wurde hier bewusst weggelassen, da dies nicht direkt Teil der Idee eines Mosaik-Plots ist.
Quelle: `https://rkabacoff.github.io/datavis/Models.html`

Stärken

- effiziente Nutzung des Raums
- skaliert für beliebig viele Daten (da die Anzahl angezeigter Flächen nur von der Anzahl Kategorien und Werte in diesen abhängt)
- Abhängigkeit des zweiten vom ersten Attribut gut sichtbar
- Abhängigkeit des $(i-1)$-ten vom i-ten Attribut einigermaßen gut sichtbar

Schwächen

- Abhängig von der Reihenfolge der Attribute
- Abhängig von der Anordnung der Kategorien
- schwere Lesbarkeit bei mehreren Variablen
- allgemein schlecht vergleichbar und lesbar
- Nullwerte müssen gesondert behandelt werden, da die entsprechende Kachel sonst einfach verschwindet

KV-Map

Die *KV-Map* ist eine fortgeschrittene Visualisierungstechnik, die zunächst – wie bei einem Mosaik-Plot – die Daten rekursiv einteilt. Dabei bleibt die Größe der Kacheln jedoch gleich verteilt und die Häufigkeit wird auf die Farbe abgebildet (entweder direkt die Anzahl Elemente oder ob die Attribute statistisch unabhängig sind).

Stärken

- geeignet für ca. 10 Attribute
- Abhängigkeiten zwischen *allen* Attributen sichtbar
- nutzt die Fähigkeit, Frequenzen und Wiederholungen wahrzunehmen

Schwächen

- Wahrnehmung muss gelernt werden
- Muster sind nicht interpretierbar (hierzu sollten automatische Verfahren genutzt werden: Visual Analytics)

Tabelle

Eine ganz allgemeine Form der Visualisierung ist eine Tabelle, in der Spalten für Attribute und Zeilen für Items verwendet werden. Dabei werden die Werte grafisch und nicht durch Text dargestellt, bspw. durch Farbe zur Indikation ob ein Wert „hoch“ ist. Es können allerdings sehr vielfältige „klein“ Visualisierungen verwendet werden.

Stärken

- Tabellen sind vertraut
- geeignet für 20 und mehr Attribute (als Heatmap)
- im Extremfall ein Datensatz pro Pixelzeile
- Attribute können kategorisch oder quantitativ sein
- Sortierung offenbart Korrelationen über viele Attribute

Schwächen

- limitiert auf mehrere hundert Datensätze

3.2 Große Datenmengen

Dieser Abschnitt behandelt spezialisierte Methoden zur Darstellung „vieler“ Daten. Wie bereits bei hochdimensionalen ist die Bedeutung „viel“ relativ zu sein: Bei der mentalen Verarbeitung kann dies bereits in der Größenordnung von wenigen Datenpunkten („mehr Daten, als man verstehen kann“), bei High-Performance-computing bei vielen Millionen Datenpunkten („mehr Daten, als man verarbeiten/speichern kann“) zutreffen. Dieser Abschnitt behandelt vornehmlich „mehr Daten, als man Anzeigen kann“, während „normale“ Visualisierungstechniken in den Bereich „mehr Daten, als man lesen kann“ fallen. Die Anzahl Datenpunkte liegt damit in der Größenordnung von einigen tausend bis zehntausenden Punkten.

3.2.1 Ordnen

In diesem Abschnitt werden Methoden behandelt, die unter dem Begriff „Ordnen“ zusammengefasst werden. Dies sind bspw. Dimensionsreduktion, Feature-Selektion und Sortieren. Bei der Dimensionsreduktion wird versucht, die Attribute so zu kombinieren, dass die Daten besser gruppiert/unterschieden werden können, bei der Feature-Selektion ist die zentrale Frage, welche Attribute tatsächlich wichtig sind und die Sortierung (als Interaktion und automatischer Prozess) stellt die Frage, wonach eigentlich gesucht wird.

Dimensionsreduktion

Ein Grundproblem der Visualisierung ist, dass ein Bildschirm (und ein Blatt Papier) nur zwei Dimensionen haben. Des weiteren sind Dimensionen, die über der dritten liegen, nur schwer bis gar nicht verständlich. Das zweite große Problem – der „Curse of Dimensionality“ – tritt in der Algorithmik auf, wenn je mehr Attribute zwei Items haben, desto eher unterscheiden sie sich auch in einem davon, d. h. es werden exponentiell mehr Daten benötigt, um „den Raum zu füllen“. Des weiteren werden Berechnungen sehr langsam und redundante Dimensionen verzerren die Ergebnisse (da wiederholte Informationen übergewichtet werden).

Formal ist das Problem der Dimensionsreduktion wie folgt definiert: Gegeben seien Daten $\mathcal{X}$ mit $n > 2$ Attributen $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{X}$ sowie eine Distanzfunktion $\mathrm{dist}_n(\cdot, \cdot)$, die einen Abstand/Unterschied zwischen zwei Datenpunkten in n Dimensionen berechnen kann. Dann wird in der Dimensionsreduktion eine Abbildung $reduce : \mathcal{X} \to \mathbb{R}^k$ gesucht, sodass $\mathrm{dist}_k(\boldsymbol{x}, \boldsymbol{y})$ sich wie $\mathrm{dist}_n(\boldsymbol{x}, \boldsymbol{y})$ verhält (für alle $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$). In der Visualisierung wird dabei oft $k = 2$ und $\mathrm{dist}_2(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2$ gewählt.

Im Folgenden werden einige übliche Verfahren zur Dimensionsreduktion angeschnitten. Eine Übersicht ist in Tabelle 3.2 zu finden.

Hauptkomponentenanalyse (Principal Component Analysis, PCA) Die *Hauptkomponentenanalyse* eignet sich gut für Daten mit linearen Abhängigkeiten, kann aber mittels Kernel-PCA auch auf nichtlineare Zusammenhänge erweitert werden. Die Reduktionsfunktion wird dabei durch Maximierung der Varianz entlang der *Hauptkomponenten* gefunden, welche über die Kovarianzmatrix berechnet werden können, sofern die Achsen im Ursprungsraum orthogonal zueinander stehen. Durch die Varianz entlang der einzelnen Achsen kann dann der Informationsverlust als *erklärte Varianz* quantifiziert werden. Bei PCA handelt es sich also um eine „verlustbehaftete Kompression“.

Linear Discriminant Analysis (LDA) *Linear Discriminant Analysis* benötigt in Klassen geteilte („gelabelte“) Daten und es werden Achsen gesucht, die die Klassen gut unterscheiden, d. h. es wird die *Reinheit* der Klassen bewertet. Dabei sind die neuen Achsen Linearkombinationen der alten Achsen. Eine übliche Methode ist

Verfahren	Besonderheiten und Stärken	Schwächen
PCA	lineares Verfahren erhält die Varianz im Originalraum einfache, geschlossene Lösung	keine nichtlineare Zsmh.
LDA	lineares Verfahren benötigt gelabelte Daten sucht Achsen, die die Labels möglichst gut trennen	keine nichtlineare Zsmh.
MDS	nichtlineares Verfahren benötigt nur Distanzen, keine Datenpunkte	Lösung wird nur approximiert
SOM	nichtlineares Verfahren auch zur Gruppierung der Daten geeignet	Lösung wird nur approximiert relativ viele Parameter

Tabelle 3.2: Übersicht über verschiedene Dimensionsreduktionsverfahren

die Optimierung des *Fisher-Kriteriums*, welches dafür sorgt, dass die Mittelwerte der Klassen maximal weit auseinander und die Varianzen innerhalb der Klassen möglichst klein werden.

Multidimensional Scaling (MDS) Bei *Multidimensional Scaling* werden die Daten so in einem niedrigdimensionalen Raum angeordnet, dass die Distanzen aus dem Originalraum möglichst gut abgebildet sind. Dies wird durch direkte Minimierung des Fehlers

$$\sum_{i,j=1}^{n} \big(d_n(\boldsymbol{x}_i, \boldsymbol{x}_j) - d_k\big(\mathrm{reduce}(\boldsymbol{x}_i), \mathrm{reduce}(\boldsymbol{x}_i)\big)\big)$$

erreicht. Die Methode funktioniert daher auch mit Daten, bei denen ausschließlich die Entfernungen bekannt sind und ist, mit Modifikationen, auch auf nominale Daten anwendbar. Die Transformation kann dabei beliebig komplex (auch nichtlinear) sein.

Self-Organizing Map (SOM) Eine *Self-Organizing Map* (SOM) ist ein Verfahren zur automatischen Anordnung von Visualisierungen (häufig Zeitserien) in einem Raster, dessen Vorgehen analog zu k-Means ist:

1. Jede Zelle (in einem fixen Raster) wird zufällig initialisiert. Diese Zellenwerte (die *Prototypen*) entsprechen den Zentroiden bei k-Means.
2. Jede Zeitserie wird der Zelle mit dem ähnlichsten (bspw. Euklidische Distanz) Prototyp zugeordnet.
3. Die Prototypen einer Zelle sowie ihre direkten Nachbarn werden so verändert, dass sie den Originaldaten, die der Zelle zugeordnet wurden, ähnlicher werden. Ferne Nachbarn (zweiten Grades) werden so verändert, dass sie den Originaldaten unähnlicher werden.
4. Schritt 2 und 3 werden so lange wiederholt, bis das Verfahren konvergiert, wodurch die Nachbarschaft in Schritt 3 immer kleiner wird.

Feature-Selektion

Ähnlich wie in der Dimensionsreduktion kann die *Feature-Selektion* formal definiert werden: Seien $\mathcal{X}$ wieder die Daten, dann wählt die Feature-Selektion davon $k < n$ *relevante* Attribute aus. Dabei ist *relevant* so definiert,

	Kontinuierliche Änderung	Diskrete Messungen
Einzelne Werte	Linien	Punkte
Anhäufung/Zuwachs	Flächen	Balken

Tabelle 3.3: Interpretation verschiedener Marks bei Zeitreihen

dass eine Vorhersage über eine (unbekannte) Eigenschaft eines Datenpunktes basierend auf den reduzierten Features das gleiche Ergebnis liefern soll wie eine Vorhersage auf den originalen Features. Die Relevanz hängt also davon ab, welche Frage gestellt wird.

Weitere Details zu Feature-Selektionsverfahren werden im Visual Analytics-Teil behandelt, siehe .

3.2.2 Aggregieren

Eine weitere Gruppe an Techniken für den Umgang mit vielen Daten ist die Aggregation. Diese basiert auf zwei Annahmen: Erstens wird ein Verfahren benötigt, welches beliebig viele Daten darstellen kann; Zweitens wird die Annahme getroffen, dass der*die Nutzer*in nicht an einzelnen Werten, sondern nur an zusammengefassten Informationen, interessiert ist. Basierend darauf gibt es drei (unabhängige) Fragestellungen bei der Aggregation, die allesamt im Kontext der Aufgabe zu sehen sind:

1. *Welche* Items werden zusammengefasst?
   Im Allgemeinen kann jede Unterteilung prinzipiell eine sinnvolle Unterteilung liefern.

2. *Wie* werden die Items zusammengefasst?
   Beispielsweise Zählen der Items, Durchschnittswert eines Attributs, häufigster Wert, Varianz eines Attributs, minimale Differenz zweier Attribute, uvm., wobei Anzahl und Durchschnitt die häufigsten Methoden sind.

3. Wie wird das *dargestellt?*
   Hier gibt es so viele Möglichkeiten wie es Möglichkeiten zur Visualisierung gibt, die Aggregation stellt nur eine Abstraktion dar. Denn: Durch die Aggregation wurde nur ein neuer Datenpunkt, bzw. neue Items, erstellt, der auf beliebige Weise visualisiert werden kann. Es ist für eine Visualisierung also egal, woher die Items kommen.

3.3 Zeitbasierte Daten

Nachdem in den vorherigen Abschnitten die Visualisierung von hochdimensionalen Daten und großen Datenmengen behandelt wurde, werden in diesem Abschnitt spezielle Techniken zur Darstellung (vieler) Zeitreihen behandelt. *Zeitbezogene Daten* sind dabei Daten, bei denen mindestens ein Key einen Zeitstempel enthält, welcher entweder absolut oder relativ ist. Da die Zeit meistens ein unabhängiges Attribut ist, wird die (fast) immer auf die Position, z. B. die horizontale Achse, abgebildet. Dabei bestimmten die Wahl der Marks die Interpretation der Daten, siehe Tabelle 3.3.

Eine Übersicht über alle in diesem Abschnitt behandelten Verfahren ist in Abbildung 3.7 zu finden.

3.3.1 Viele Zeitreihen

Zur Darstellung vieler Zeitreihen die, wenn sie alle als Linie in einer Visualisierung gezeichnet werden würden, zu Overplotting führen, gibt es verschiedene Optionen, die hier behandelt werden.

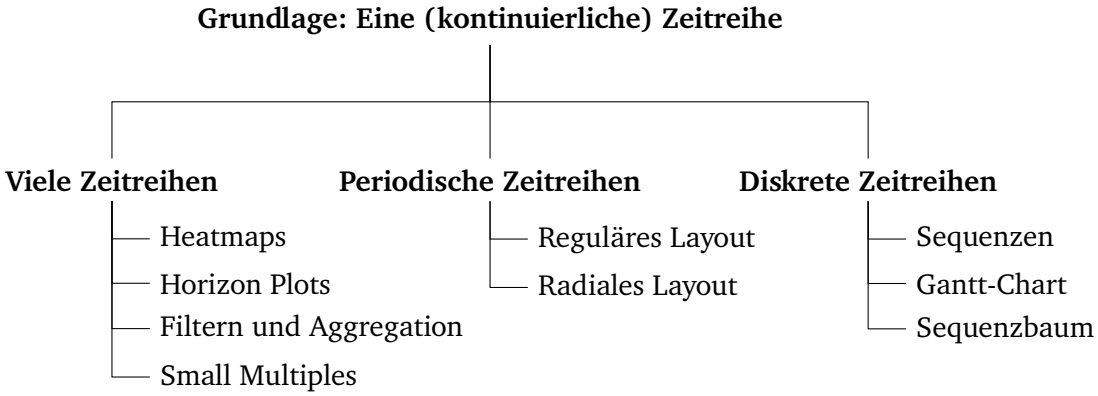

Abbildung 3.7: Übersicht über verschiedene Zeitreihendarstellungen

Filtern

Die einfachste Option zur Darstellung vieler Zeitreihen ist die Filterung, bspw. durch Interaktives Brushing. Dabei kann z. B. ein Wertebereich markiert werden und es werden nur Daten angezeigt, die durch diesen Bereich laufen.

Heatmaps

Die Darstellung von Zeitreihen als *Heatmap* (mit der Zeit in der horizontalen Position und den Werten farblich dargestellt) lassen sich sehr viele Zeitreihen kompakt darstellen, da pro Zeile nur eine Pixelreihe benötigt wird. Die Daten werden also im Allgemeinen nicht aggregiert, sondern nur sehr kompakt dargestellt.

Stärken

- sehr kompakte Darstellung vieler Zeitreihen
- gute Vergleichbarkeit von Trends entlang der Zeitachse

Schwächen

- keine gute Vergleichbarkeit von Werten

Horizon Plots

Ein *Horizon Plot* ist eine Mischung aus Liniendiagramm und Heatmap, bei denen die Werte sowohl auf die Farbe als auch die Höhe abgebildet werden. Dabei wird die Höhenachse jedoch für verschiedene Skalen genutzt, die durch die Farben einer divergierenden Farbskala gekennzeichnet werden (siehe Abbildung 3.8).

Stärken

- sehr kompakte Darstellung vieler Zeitreihen (etwas 50 bis 100)
- bessere Vergleichbarkeit von Werten als bei Heatmaps

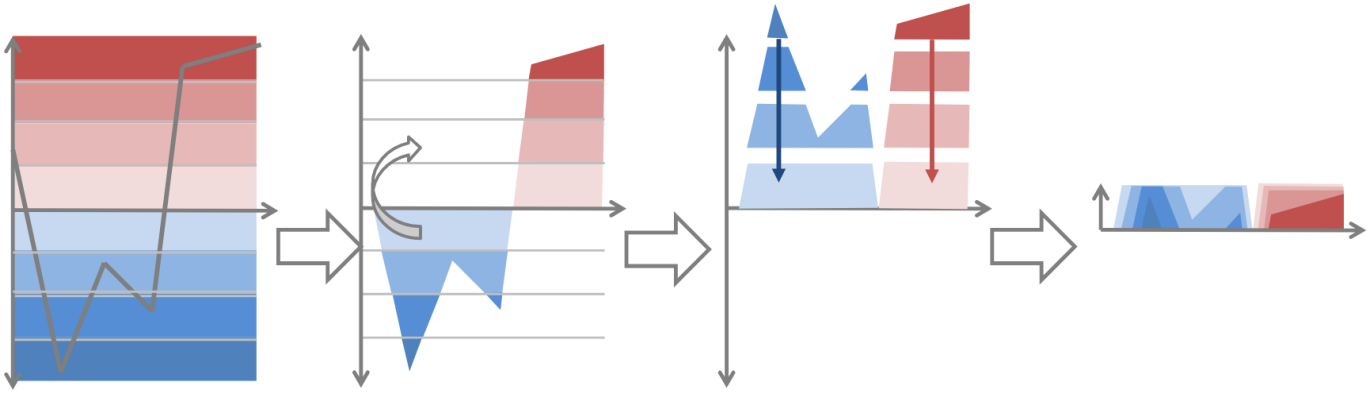

Abbildung 3.8: Visualisierungstechnik (viele Zeitreihen): Horizon Plot
Quelle: Vorlesungsfolien

Schwächen

- konkrete Werte sind komplex abzulesen

Heatmaps mit Aggregation

Bei einer Heatmap mit Aggregation wird jeder Pixel einer Ebene (mit Zeit- und Werteachse) durch eine „Dichte“ der Linien bestimmt, die durch den jeweiligen Punkt gehen würden (*Density Heatmap*). Dadurch werden viele Zeitreihen auf wenige Details reduziert und es können Trends erkannt werden.

Small Multiples

Wie bereits bei der Visualisierung von hochdimensionalen Daten können auch hier Small Multiples eingesetzt werden, indem die Zeitreihen in einem Raster angeordnet werden. Dadurch leidet allerdings die Vergleichbarkeit, da nur benachbarte Reihen gut vergleichbar sind – außerdem führen horizontale und vertikale vergleiche zu verschiedenen Ergebnisse, da einmal Werte und einmal Zeitverläufe verglichen werden. Eine Abhilfe schafft hier die Aggregation durch Self-Organizing Maps (siehe Abschnitt 3.2.1), die die Zeitreihen gruppiert und zusammenfasst. Dies erfordert jedoch irgendeine Form von Cluster, was sehr aufwendig sein kann. Dafür skaliert der Ansatz allerdings auf deutlich mehr Zeitreihen.

3.3.2 Periodische Zeitreihen

Eine spezielle Art von Zeitreihen sind *periodische*, bei denen sich einige Abläufe wiederholen (z. B. Jahre, Monate, Woche, Pulsschläge, Schall, etc.). Das Ziel ist dabei eine Visualisierung über die Zyklen hinweg. Dabei entstehen die Herausforderungen, die Periodenlänge zu bestimmen sofern diese nicht bekannt ist und mit variablen Frequenzen umzugehen (bspw. Puls). Die folgenden Ansätze dienen der reinen Visualisierung unter der Annahme, dass die Periodenlänge λ bereits bekannt ist.

Spirallayout

Spirallayouts „rollen“ die Zeitachse auf, d. h. die Daten werden spiralförmig aufgetragen (siehe Abbildung 3.9). Durch Variation der Periodenlänge λ können, sobald die Visualisierung „einrastet“, Muster gefunden werden. Zur Erstellung der Grafik werden Polarkoordinaten $r = \tilde{r}t/\lambda$, $\varphi = 2\pi t/\lambda$ mit einem Radius $\tilde{r}$ genutzt.

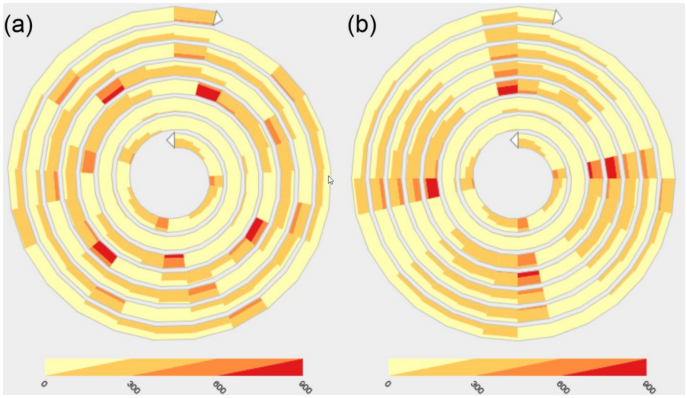

Abbildung 3.9: Visualisierungstechnik (periodische Zeitreihen): Spirallayout
Quelle: Vorlesungsfolien

Stärken

- kompakt (wie die meisten radialen Layouts)

Schwächen

- die Zeitachse wird nach außen breiter; Verzerrung
- funktioniert nur bedingt als Liniengrafik

Matrix-Layout

Bei einem *Matrix-Layout* werden die einzelnen Zeitreihen gestapelt, basierend auf dem Modulo-Operator:

$$x = t \mathbf{mod} \lambda \qquad\qquad y = \lfloor t/\lambda \rfloor$$

Wie bei einem Spirallayout können Muster durch Variation von λ und visuellem „Einrasten“ gefunden werden. Dieses Layout funktioniert im Prinzip auch mit Liniendiagrammen, ist aber häufig als Heatmap anzutreffen.

3.3.3 Diskrete Ereignisse

Bei *Ereignissen* haben diese zwar einen Zeitstempel, aber nicht unbedingt ein quantitatives Attribut, welches auf eine y-Achse abbildbar wäre[2], d. h. die Ereignisse sind Dimensionslose Punkte im Raum (bspw. Deadlines). Bei der Darstellung gibt es dann viel Gestaltungsspielraum abhängig davon, was relevant ist: die Reihenfolge der Ereignisse oder die Dauer zwischen diesen:

- Ist die Reihenfolge das primär interessante, können die Ereignisse als Sequenz ohne Abstand dargestellt werden.
- Ist jedoch der Abstand zwischen den Ereignissen interessant, so können diese als Gantt-Chart dargestellt werden.

[2]Eine kontinuierliche, „normal“, Reitreihe kann durch Diskretisierung in eine Serie diskreter Ereignisse zerlegt werden indem die x- und y-Achse in einzelne Bereiche geteilt werden.

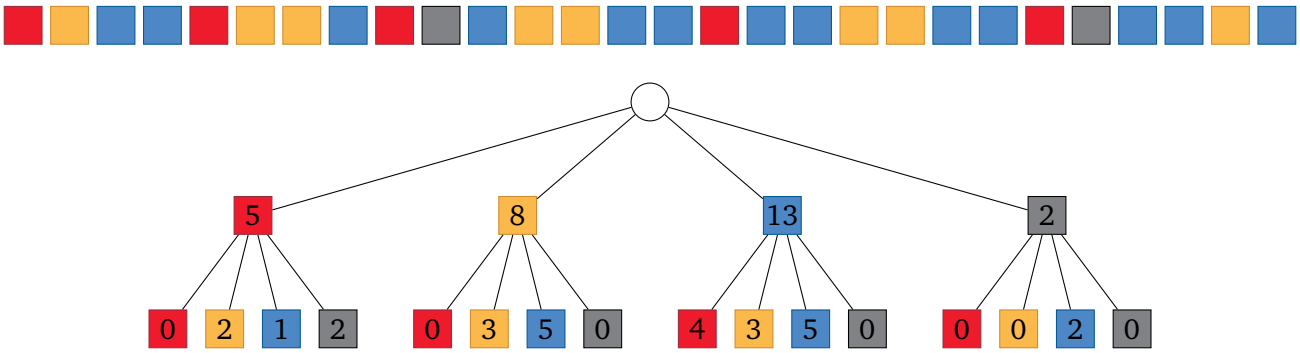

Abbildung 3.10: Beispiel für einen Sequenzbaum. Die Zahl eines Knotens repräsentiert, wie oft das Muster, welches durch verfolgen vom Wurzelknoten zu diesem Knoten entsteht, vorkommt. Die Summe der Zahlen der Kinder eines Knotens muss damit immer der Zahl in diesem Knoten entsprechen (per Definition).

Eine andere Herangehensweise stellt ein *Sequenzbaum* dar: Dabei wird eine Zeitreihe in wiederkehrende Muster basierend auf der Reihenfolge zerlegt. Dabei entspricht die Ebene des Baums (wobei der Wurzelknoten auf ebene 0 ist) der Länge der Sequenz, die betrachtet wird. In jedem Knoten wird dann die Anzahl der Sequenzen geschrieben, die dem Pfad vom Wurzelknoten bis zu dem Knoten entsprechen. Dies ist am besten an einem Beispiel zu verstehen, siehe dafür Abbildung 3.10. Als alternative können die Zahlen in den Knoten auch weg gelassen werden und die Anzahl durch die Breite der Kanten repräsentiert werden, was den Sequenzbaum kompakter macht.

3.4 Graphen und Bäume

In diesem Abschnitt wird die Visualisierung von Graphen und, als Spezialfall, von Bäumen behandelt. Insbesondere Graphen sind sehr komplexe Datenstrukturen, die sich auch einer Knotenmenge $\mathcal{V}$ und einer Kantenmenge $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ zusammensetzen. Dabei können sowohl Kanten als auch Knoten weitere Attribute enthalten; außerdem können die Knoten implizit durch die Kantenliste gegeben sein. Zur weiteren Behandlung von Visualisierungen müssen zunächst einige Begriffe definiert werden:

- Eine Folge von Knoten ist ein *Weg* gdw. aufeinander folgende Knoten durch Kanten verbunden sind.
- Ein Graph ist *zusammenhängend* gdw. zwischen je zwei Knoten ein Weg existiert.
- Ein Graph ist *gerichtet* gdw. die Kanten nicht symmetrisch sind, d. h. $(v_1, v_2) \in \mathcal{E} \not\Longleftrightarrow (v_2, v_1) \in \mathcal{E}$.
- Ein Graph ist *azyklisch* gdw. es keinen Weg gibt, der einen Knoten mehrfach erreicht.
- Ein *Baum* ist ein azyklischer, zusammenhängender Graph.

Bei der Visualisierung kann dann im Allgemeinen gesagt werden, dass die Visualisierung eines Baumes leicht, die eines gerichteten, azyklischen Graphen mittel bis leicht und die eines allgemeinen Graphs extrem schwer ist. Dies basiert darauf, dass die ersten beiden Graphen eine *topologische Ordnung* beschreiben.

Eine Übersicht über die vorgestellten Verfahren ist in Abbildung 3.11 zu finden.

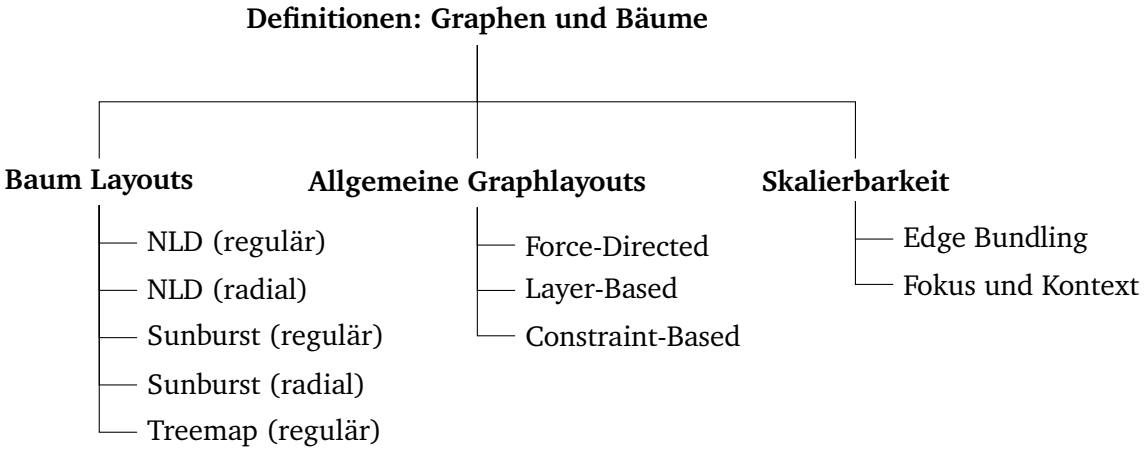

Abbildung 3.11: Übersicht über verschiedene Graph- und Baumdarstellungen; Die Abkürzung „NLD“ steht für „Node-Link-Diagramm“.

3.4.1 Bäume

Im Folgenden wird angenommen, dass jeder Baum einen *Wurzelknoten* v_1 ohne Eltern besitzt[3].

Node-Link-Diagramm

Die einfachste Visualisierung ist ein *Node-Link-Diagramm*, in dem Knoten als Punkt und Kanten als Linien dargestellt werden. Der Abstand zur Wurzel definiert dabei die vertikale Position (siehe Abbildung 3.12). Die Kanten werden dabei explizit dargestellt und es ist immer möglich, ein kreuzungsfreies (*planares*) Layout zu finden. Ein offenes Problem ist eine „gute“ Positionierung innerhalb einer Ebene. Des weiteren wird das Diagramm bei vielen Kindknoten sehr breit. Einige Gütekriterien für ein gut strukturiertes Layout sind:

- keine Kreuzungen
- alle Knoten einer Hierarchiestufe auf gleicher Höhe
- möglichst schmal (um den Platz einer Seite gut auszunutzen)
- Elternknoten zentriert über den Kindknoten
- Symmetrien sollten erkennbar sein
- Ordnungserhaltend
- erstellbar in linearer Zeit

Im Allgemeinen sind nicht alle dieser Anforderungen gleichzeitig erfüllbar. Zur Optimierung der Kriterien können bspw. Knoten verschoben oder Teilbäume vertauscht werden.

[3] In jedem Baum gibt es aufgrund der Azyklizität mindestens einen solchen Knoten.

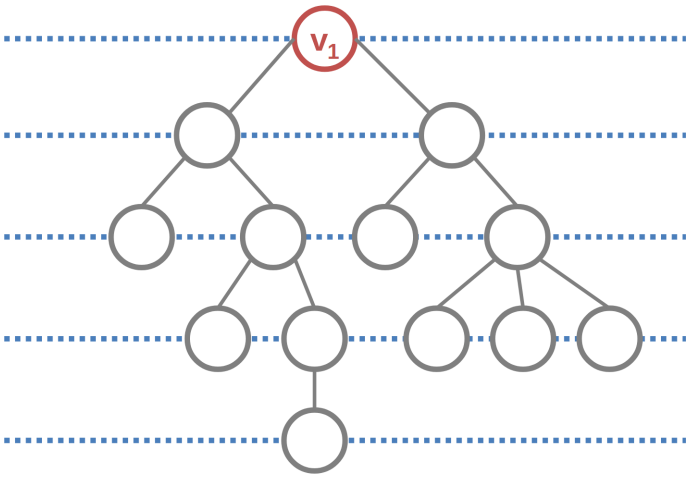

Abbildung 3.12: Visualisierungstechnik (Bäume): Node-Link-Diagramm
Quelle: Vorlesungsfolien

Radiales Node-Link-Diagramm

Eine Option dem Problem der Breite bei einem Node-Link-Diagramm entgegen zu wirken ist, dieses „aufzurollen“. Dabei wird der Wurzelknoten in der Mitte platziert und die Baumtiefe wird durch den Radius abgebildet. Dadurch wird die Breite limitiert, allerdings kann es schwer sein, das Gesamtbild zu erfassen.

TreeMaps

Bei *TreeMaps* wird die Beziehung zwischen Knoten dargestellt als eine „enthalten in“-Beziehung. Dabei wird der Wurzelknoten repräsentiert durch die Gesamtfläche des Bildes und die Kindknoten werden, ähnlich wie bei einem Mosaik-Plot (siehe **??**), durch Teilung des Raums dargestellt (siehe Abbildung 3.13). Die Kanten werden also nicht explizit dargestellt, sondern der Fokus wird auf die Struktur gelegt. Eine Option ist, den Flächeninhalt der Knoten gleichmäßig oder anhand bestimmter Eigenschaften des Knotens (z. B. Dateigröße oder Anzahl Kindknoten) zu bestimmen.

Um die Linien zur Abgrenzung der Knoten zu reduzieren, können *Cushions* verwendet werden. Das sind Farbverläufe die, wenn man sie nebeneinander Setzt, eine Abgrenzung sichtbar werden lassen. Dadurch müssen keine expliziten Kanten mehr gezeichnet werden. Eine weitere Variante von TreeMaps sind *Squarified* TreeMaps, in auf der gleichen Hierarchiestufe sowohl vertikal und horizontal geteilt wird (abhängig davon, wie Platz ist). Dadurch ist die TreeMap besser lesbar und entartete Rechtecke werden vermieden, allerdings ist die direkte Eltern-Kind-Beziehung nicht mehr eindeutig.

Icicle-Plots und Sunburst

Icicle-Plots und *Sunbursts* sind, ähnlich wie TreeMaps, raumfüllende Techniken, bei denen die Knotengrößen über Flächen dargestellt werden. Bei einem Icicle-Plot werden, wie bei einer TreeMap, die Kanten nicht explizit dargestellt, sondern die Kindknoten unter dem jeweiligen Elternknoten platziert. Dabei nimmt der Wurzelknoten die gesamte Breite ein und die Kindknoten werden immer kleiner. Ein Sunburst ist dann genau das gleiche, nur aufgerollt (Abbildung 3.14).

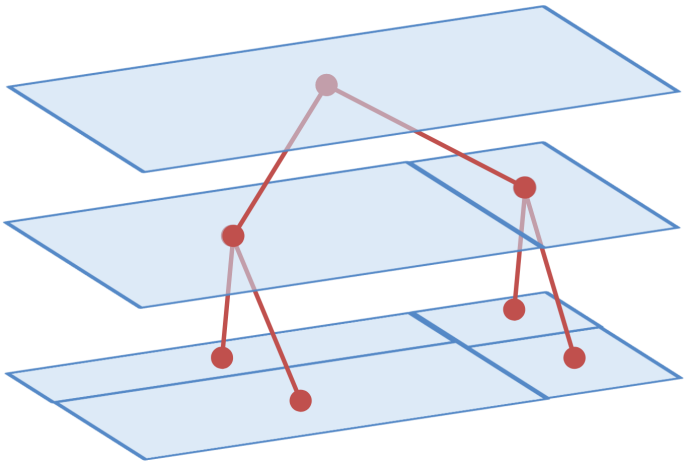

Abbildung 3.13: Visualisierungstechnik (Bäume): TreeMaps; Das enthaltene Node-Link-Diagramm (blau) dient nur der Illustration. Die TreeMap ist die untere blaue Ebene.
Quelle: Vorlesungsfolien

(a) Icicle-Plot

(b) Sunburst

Abbildung 3.14: Visualisierungstechnik (Bäume): Icicle-Plot und Sunbursts; Die enthaltenen Node-Link-Diagramme dienen nur der Illustration und sind nicht Teil des jeweiligen Plots.
Quelle: Vorlesungsfolien

3.4.2 Allgemeine Graphen

Während im vorherigen Abschnitt das „einfache“ Problem der Baumvisualisierung behandelt wurde, geht es in diesem Abschnitt um die Visualisierung allgemeiner Graphen und damit um ein sehr viel schwereres Problem.

Node-Link-Diagramm Grundlegend bauen die meisten der folgenden Visualisierungen auf einem einfachen Node-Link-Diagramm auf, in dem die Knoten durch Punkte und die Kanten durch Striche dargestellt werden. Das große Problem – welches von verschiedenen Verfahren auf unterschiedlichste Arten gelöst wird – ist das Layouting der Kanten und Knoten.

Stärken

- die Pfade sind nachverfolgbar
- gerichtete Kanten sind explizit darstellbar (z. B. als Pfeile)
- im Prinzip für bis zu $10\,000$ Knoten brauchbar

Schwächen

- zwei Layouting-Dimensionen (Position)
- Graphen mit vielen Kanten (dichte Graphen) sind kaum lesbar

Adjazenzmatrix Eine andere Kategorie an Visualisierungen baut auf der *Adjazenzmatrix* des Graphen auf. Dabei werden die Knoten über beide Achsen verteilt und die Kanten in der entstehenden Matrix markiert.

Stärken

- keine Überlappungen
- dichte Graphen sind gut darstellbar
- gerichtete Kanten erzeugen Asymmetrie und sind somit gut sichtbar
- „nur“ eine Layouting-Dimension (Sortierung)

Schwächen

- Verfolgung von Pfaden ist schwer
- dünn besetzte Graphen nutzen den Platz nicht aus
- Sortierung beeinflusst die Struktur
- „nur“ für wenige hundert Knoten nutzbar

Layouts

Wie bereits erwähnt ist das größte Problem von Node-Link-Diagrammen das Layouting, also das finden eines Layouts, welches die folgenden Anforderungen erfüllen sollte:

- Vermeidung unnötiger Kantenüberschneidungen (idealerweise planare Darstellung)
- übereinander liegenden Knoten sollten vermieden werden (Overplotting)
- verbundene Knoten sollten nah beieinander sein, d. h. die Kanten sollten kurz sein
- stark verbundene Teilgraphen sollten gut erkennbar sein
- der gegebene Platz sollte ausgenutzt werden
- benachbarte Knoten sollten im Durchschnitt gleich weit voneinander entfernt sein

Die bereits erwähnt ist das Layouting sehr schwierig, wenn nicht sogar das schwerste (algorithmische) Visualisierungsproblem, welches im Allgemeinen noch ungelöst ist. Die im folgenden vorgestellten Verfahren liefern dementsprechend nur Approximationen und keine exakte Lösungen.

Force-Directed Eines der ersten vorgestellten Verfahren ist das *Force-Directed* Layout. Dabei werden die Knoten und Kanten als Feder-Masse System mit zusätzlichen abstoßenden Kräften, wenn zwei Knoten nicht verbunden sind. Die anziehenden Kräfte nehmen dabei linear mit dem Abstand zu während die Abstoßenden Kräfte quadratisch abnehmen. Durch eine Physiksimulation der Kräfte und Bewegungen entsteht dann ein Layout gemäß dieser Kräfte. Sobald sich keine Knoten mehr bewegen, ist das Layout fertig.

Stärken

- sehr einfach, es ist kein Wissen über den Graph notwendig
- erlaubt Interaktion (z. B. Verschiebung der Knoten)
- Kanten und Knoten können gewichtet werden

Schwächen

- lange Laufzeit bei vielen Knoten und Kanten
- keine optimal Lösung

Layer-Based (Sugiyama) Ein *Layer-Based* Verfahren eignet sich für gerichtete, azyklische Graphen und ist durch „Entfernung“ von Schleifen oder eine künstlicher „Richtung“ des Graphen auch auf allgemeine Graphen anwendbar. In einem ersten Schritt werden die Kanten entsprechend ihrer Entfernung zu den „Wurzelknoten“ (den Knoten ohne Eltern) von links nach rechts in Schichten einsortiert. Dabei bleibt die Höhe eines Knoten in seiner Spalte zunächst unbestimmt. Das Sugiyama-Layout nutzt im zweiten Schritt (dem Zuordnen einer Höhe) einen einfachen lokalen Greedy-Algorithmus. In jeder Iteration werden einfache Modifikationen (z. B. Vertauschung von zwei Knoten) in einer Spalte durchgeführt, evaluiert und die beste beibehalten. Sobald keine Verbesserung mehr möglich ist, fährt der Algorithmus mit der nächsten Spalte fort. Es existieren auch komplexe Backtracking-basierte Algorithmen, die hier nicht diskutiert werden.

Stärken

- einfaches Verfahren
- strukturiert (azyklische) Graphen
- definiert eine Hauptrichtung

Schwächen

- Graphen mit (wenigen) Zyklen müssen gesondert behandelt werden
- nicht brauchbar für Graphen mit vielen Zyklen

Constraint-Based (Metro-Map) Bei einem *Constraint-Based* Layout werden bestimmte Bedingungen gestellt, innerhalb derer der Graph angeordnet wird. Das bekannteste Beispiel solcher Layouts sind *Metro-Maps* (vgl. Liniennetzpläne). Die charakteristischen Merkmale eines solchen Layouts sind:

- Knoten sind auf einem Gitter angeordnet
- Kanten verlaufen nur in bestimmten Winkeln
- Kantenkreuzungen sind häufig Knoten
- Knotenform ist an die Kreuzungsform angepasst

Dabei übernimmt das Verfahren einige Ideen des Sugiyama-Layouts: Für die Qualität des Layouts werden, abgeleitet aus den obigen Merkmalen, einige Kriterien festgelegt und ein Layout danach bewertet. Durch eine iterative lokale Verbesserung konvergiert das Verfahren dann gegen ein lokales Optimum.

Stärken

- Anwendbar auf allgemeine Graphen, aber das Knoten-Kanten-Verhältnis sollte „ausgewogen“ sein
- visuell stark strukturiert und dadurch (vergleichsweise) einfach zu lesen

Schwächen

- teilweise komplexe Heuristiken
- nur lokal optimale Lösungen
- „Pfade“ müssen ggf. definiert werden

(Hierarchisches) Edge-Bundling

Ein großes Problem bei vielen Layouts ist nicht-Lesbarkeit durch zu viele Kanten. Dem kann entgegengewirkt werden indem „ähnliche“ Kanten, also Kanten, die aus einem Knotencluster in einen anderen laufen, gebündelt werden. Dabei werden die Kanten nicht mehr als kürzeste Wege zwischen Knoten gezeichnet, sind dadurch aber besser lesbar. Eine Erweiterung dieser Idee stellt *hierarchisches* Edge-Bundling dar: Die Grundidee ist, dass die Knoten innerhalb einer Hierarchie liegen und eine Kante statt des direkten Weges über die Hierarchie zum gemeinsamen Vorfahren läuft. Dadurch durchlaufen mehrere Kanten den gleichen Pfad, was eine Bündelung ergibt.

„Search, Show Context, Expand on Demand“

Die Grundidee von „Search, Show Context, Expand on Demand“ ist, nicht direkt den ganzen Graph anzuzeigen. Stattdessen werden nur die wichtigsten Knoten sowie deren Nachbarschaft angezeigt, wobei sich die Wichtigkeit daraus ergibt, welche Knoten zuletzt angeklickt oder in einer Suche gefunden wurden. Zu Beginn müssen dafür die Knoten auf irgendeine Weise sinnvoll gefiltert werden. Dadurch wird das Layout auf die Aufgabe vereinfacht und der Fokus verschiebt sich auf die relevanten Knoten. Verbindungen zu weiteren, nicht dargestellten, Knoten können dabei als Verbindung in den „leeren Raum“ dargestellt werden.

3.5 Geobasierte Daten und Karten

3.5.1 Karten als Metapher

Karten und Schematisierungen

3.5.2 Geobezogene Daten

Kartenprojektion

Plattkarte

Mercator Projektion

Winkel-Tripel

Verzerrte Darstellungen

Metro-Map

(Stetige) Kartogramme

Abstrakte Geovisualisierungen

3.5.3 Nicht-Geobezogene Daten

Wikipedia World Map

Themescapes

Themengebiete

Gmap World of Music

Metro-Map Immitation

Rekonstruktion von Terrain aus Knotenattribut

3.5.4 Raum-Zeit Daten

Darstellung von Richtungen

Darstellung von Geschwindigkeiten

Darstellung von Vielen Trajektorien