

MASTERARBEIT

Entwicklung eines R-Pakets für Data Storytelling und Datenvisualisierungen

Autor

Andriu CAVELTI
16-057-390
andriu.cavelti@stud.unibas.ch
Dorfstrasse 30, 6005 Luzern

Referent

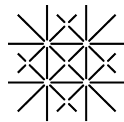
Prof. Dr. Lukas ROSENTHALER

Koreferentin

Dr. Vera CHIQUET

Herbstsemester 2022

Abgabedatum: 17. Februar 2023



**Universität
Basel**

Inhaltsverzeichnis

1	Einleitung	5
2	Von Datenwerten zur Visualisierung	8
2.1	Das kartesische Koordinatensystem als Positionsskala	10
2.2	Farbe als Steuerungsinstrument	12
2.3	Gestaltprinzipien in Datenvisualisierungen	14
2.4	Mit Daten Geschichten erzählen	17
3	Vorgehen und Methode	20
3.1	Tidy Data als Basis von Datenvisualisierungen	20
3.2	Die Beziehung von Daten und Visualisierungen mit Grammar of Graphics .	21
4	R-Paket Entwicklung	26
4.1	Struktur im Prozess	26
4.2	Tidy evaluation	27
5	Workflow für die Entwicklung von Paketen in R	30
5.1	Schritt 1: Initiierung des Pakets	30
5.2	Schritt 2: Funktionen schreiben	31
5.3	Schritt 3: Dokumentation	31
5.4	Schritt 4: Testen	33
5.5	Schritt 5: Installieren	33
6	Verwendung von {biviz}	34
6.1	Datenvisualisierungsfamilien	34
6.2	Vorteile	35
7	Fazit	38

Abbildungsverzeichnis

2.1	Skalen verknüpfen Datenwerte mit Aesthetics. Quelle: Wilke (2020), S. 10. .	10
2.2	Anzahl brennbare Abfälle und Sperrgut je Gemeinde im Jahr 2021. Eigene Darstellung.	11
2.3	Die Anzahl Arbeitslose sind in Tausend angegeben. Datenquelle: Teil des ggplot2 Pakets. Eigene Darstellung.	11
2.4	Zahlen mit und ohne präattentivem Merkmal. Quelle: (Nussbaumer Knaflitz 2017, S. 86).	13
2.5	Das R Paket colorspace (Zeileis u. a. 2020) ist ein flexibles Werkzeug um eigene Farbpaletten zu erstellen (beispielsweise eine Okabe Ito Farbskala) oder auf bestehende Skalen zuzugreifen. Eigene Darstellung.	14
2.6	Das Gestaltprinzip der Nähe.	15
2.7	Das Gestaltprinzip der Ähnlichkeit.	15
2.8	Das Gestaltprinzip der Umrandung.	15
2.9	Das Gestaltprinzip der Form.	16
2.10	Das Gestaltprinzip der Kontinuität.	16
2.11	Das Gestaltprinzip der Verbindung.	16
3.1	Bei einem tidy Datenset sind Variablen in Spalten, Beobachtungen in Zeilen und Werte in Zellen gespeichert. Quelle: Wickham und Grolemund (2016), S. 149.	20
3.2	Ordnung durch Tidy Data. Quelle: Lowndes und Horst (2020).	21
5.1	Initiierung des Pakets.	30
6.1	Verfeinern von biviz Visualisierungen.	37

Tabellenverzeichnis

2.1	Basis Aesthetics Quelle: Wilke (2020), S. 8.	8
2.2	Variablentypen für Visualisierungen. Quelle: Wilke (2020), S. 9. Eigene Anpassungen.	9
2.3	Abfallart und -menge im Kanton Zürich. Datenquelle: Teil des biviz Pakets.	9
2.4	Die zwei Arten der Analyse. Quelle: Dykes (2020), S. 138. Eigene Anpassungen.	17

1 Einleitung

i Hinweis

Diese Arbeit ist mit [Quarto](https://quarto.org/) (<https://quarto.org/>) geschrieben. Eine HTML-Version der Arbeit steht zusammen mit der Dokumentation auf der Website des Pakets [biviz](https://tricktracktriu.github.io/biviz/) (<https://tricktracktriu.github.io/biviz/>) zur Verfügung. Der Quellcode des Pakets ist unter [GitHub](https://github.com/tricktracktriu/biviz/) (<https://github.com/tricktracktriu/biviz/>) abrufbar.

Datenvisualisierungen und die Auseinandersetzung mit Informationsdesign für quantitative Daten gab es schon lange vor Big Data. John W. Tukey (Tukey 1977), Edward R. Tufte (Tufte 2001, Erstveröffentlichung 1982) und William S. Cleveland (Cleveland und McGill 1984) sind bis heute prägend für das Gebiet der Datenvisualisierung. Datenvisualisierungen wurden aber schon viel früher für die Erkenntnisgewinnung verwendet. John Snow, ein britischer Epidemiologe im 19. Jahrhundert, erkannte durch die Visualisierung von Choleratodesfällen auf einer Karte von London (jeder Todesfall war ein kleiner schwarzer Balken), dass die Cholerafälle sich rund um eine Wasserpumpe in der Broad Street konzentrierten. Nach der Abschaltung der Wasserpumpe gab es keine weiteren Fälle mehr (Gerste 2022, Kapitel Karte des Todes; Snow 1855, S. 45). Datenvisualisierungen können wichtige Einblicke in Daten geben. Dennoch werden sie in der statistischen Ausbildung oft vernachlässigt.

The Analysis stage has traditionally been the main emphasis of statistics courses [...] but sometimes all that is required is a useful visualization [...]. (Spiegelhalter 2020, S. 15)

Datenvisualisierungen alleine reichen aber nicht aus. Sie müssen eine Geschichte erzählen, damit sie zu einer Aktion führen, beispielsweise zur Abschaltung der Wasserpumpe in der Broad Street. Das Storytelling ist bei der Arbeit mit Daten eine der wichtigsten Aufgaben. Nate Silver beschreibt es wie folgt:

The number have no way of speaking for themselves. We speak for them. We imbue them with meaning. (Silver 2020, S. 9)

Als ich bei meinem alten Arbeitgeber angefangen habe, bemerkte ich schnell, dass diese Erkenntnisse in der Berufswelt noch nicht überall fassbar gemacht haben. Daten wurden meistens als Tabellen oder einzelne Werte in Excel oder PowerPoint dargestellt. Vereinzelt gab es auch Grafiken. Bei der Erstellung wurde aber nicht auf die gewünschte Botschaft und ihre Wirkung geachtet. Bei diesem Workflow gibt es zwei grosse Probleme, welche die Qualität der Datenanalysen einschränken. Zum einen sind die Reports oder Analysen nicht reproduzierbar. In Excel sind Copy-Paste und händische Arbeitsschritte der Standard

bei der Aufbereitung sowie Verarbeitung von Daten. Folglich können die Auswertungen nicht reproduziert werden. Das hat weiter zur Folge, dass die Arbeitsschritte bei jeder Auswertungsperiode manuell abgearbeitet werden müssen. Das kann zu Fehlern führen und bringt viel Aufwand mit sich. Leider können dabei die Fehlerquellen nicht mit Sicherheit ausgemacht werden, da die manuellen Arbeitsschritte nicht “aufgezeichnet” werden. Ein Skript basierter Workflow kann beiden Problemen entgegenwirken. Im Skript werden alle Arbeitsschritte als Code festgehalten. Am besten werden gleich ganze Datenpipelines erstellt. Bei wiederkehrenden Reports muss anschliessend nur die Datenquelle aktualisiert und das Skript laufen gelassen werden. Alle Arbeitsschritte werden anschliessend automatisch durchgeführt. Durch den Code sind die einzelnen Arbeitsschritte “verschriftlicht” und der Computer führt die Anweisungen im Skript aus. Somit kann genau nachvollzogen werden, was, wann und wie gemacht wird. Das heisst, der Report ist reproduzierbar. Das zweite Problem ist die Annahme, dass die Daten für sich selbst sprechen und Datenvisualisierungen verwendet werden, ohne sich Gedanken dazu zu machen, ob die gewählte Form nützlich für die Botschaft ist. Diese Problematik für die Qualität der Datenanalysen ist der Ausgangspunkt dieser Masterarbeit. Damit Analysen gehört werden, müssen sie überzeugen. Datenvisualisierungen sind ein wichtiges Tool, um die Daten und Erkenntnisse klar und aussagekräftig zu präsentieren. Der Masterarbeit ging die Fragestellung voraus: Wie wird Data Storytelling integraler Bestandteil eines Report? Das ist eine komplexe Frage und steht im Zusammenhang mit Arbeitsprozessen sowie Gewohnheiten. Wilke (2020) vergleicht die Fähigkeit klare, attraktive und überzeugende Visualisierungen zu erstellen mit dem “Ohr” eines Textredakteurs, der beim Lesen innerlich zu “hört”, ob ein Text gut geschrieben ist:

[...] brauchen wir in ähnlicher Weise ein “Auge”, also die Fähigkeit, eine Abbildung zu betrachten und festzustellen, ob sie ausgewogen, klar und überzeugend ist. Und ebenso wie beim Beurteilen von Text kann die Fähigkeit, zu sehen, ob eine Abbildung funktioniert oder nicht, erlernt werden. [...] Meine Erfahrung nach entwickeln Sie [...] noch kein Auge, wenn Sie am Wochenende mal ein Buch lesen. Es ist ein lebenslanger Prozess [...]. (Wilke 2020, S. XI)

Da es in einer Masterarbeit schwer ist, einen lebenslangen Prozess zu untersuchen, verfolgt diese Masterarbeit ein bescheidenes Ziel. Die Arbeit möchte die Entwicklung von Data Storys und Datenvisualisierungen unterstützen. Dies geschieht indem in R ein Visualisierungsprogramm entwickelt wird. Ziel des Pakets (Software) ist es, oft verwendete Datenvisualisierungen möglichst einfach zu erstellen. Gleichzeitig sollen die Grafiken einfach individuell weiterentwickelt werden können. Das heisst, die Funktionen im entwickelten Paket erzeugen Datenvisualisierungen, die so verwendet werden können, ihr ganzes Potential aber erst mit einigen interaktiven Codeergänzungen ausschöpfen. Die Idee dabei ist, dass die Hürde eine Datenvisualisierung zu erstellen, möglichst gering ist. Je einfacher ein Prozess ist, desto eher wird er angewendet. Es ist schade, wenn aufgrund des befürchteten Arbeitsaufwands (oder Zeitdrucks) keine Grafik erstellt wird oder verschiedene Formen der Grafik nicht getestet werden (was immer der Fall sein sollte). Mit der Zunahme der Datenmenge (Big Data) geht auch der Wunsch einher, die Daten zu verstehen. Dank der Erzählung von Geschichten mittels Datenvisualisierung, werden Daten in Informationen verwandelt, diese helfen Verständnis für eine Sachlage zu schaffen und Entscheidungen zu treffen (Nussbaumer Knaflitz 2017). Mit dieser Masterarbeit soll dieser Prozess vereinfacht werden, ohne an Flexibilität einzubüssen.

Im ersten Teil der Arbeit werden die wichtigsten Punkte für die Erstellung von Visualisierungen mit Daten eingeführt und erläutert. Wie werden aus Datenwerten Datenvisualisierungen? Zum einen sind das technische Aspekte wie das Koordinatensystem, zum anderen psychologische Faktoren wie die Verarbeitung von kognitiven Reizen. Im zweiten Teil wird auf Methoden und Voraussetzungen der Datenvisualisierungen, die bei der Erstellung des Pakets verwendet wurden, eingegangen. Der dritte Abschnitt erläutert die klassischen Entwicklungsschritte eines Pakets. Abschliessend wird die Anwendung des entwickelten Pakets beispielhaft gezeigt und ein Fazit gezogen.

2 Von Datenwerten zur Visualisierung

Daten zu visualisieren heisst, eine Transformation von Daten hin zu systematischen und logischen visuellen Elementen zu vollziehen, welche in ihrer Summe als bildliche Einheit interpretiert werden. Im Kern ordnet der Transformationsprozess den Daten ein quantifizierbares Merkmal zu. Das kann eine Grösse, eine Farbe, eine Form oder eine Position sein. In der Datenvisualisierung stehen 4 grundlegende Aesthetics (Gestaltungselemente) für die Darstellung von Daten als Grafiken zur Verfügung (Wilke 2020, S. 7).

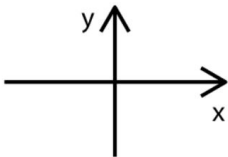



Position	Grösse	Farbe	Form
			

Tabelle 2.1: Basis Aesthetics Quelle: Wilke (2020), S. 8.

Die Aesthetics Linientyp, Linienbreite und Transparenz können als Spezialelemente von Form, Grösse und Farbe interpretiert werden.

Aesthetics werden in zwei Skalen unterteilt: solche die kontinuierliche Daten darstellen können und solche, die das nicht können (Wilke 2020, S. 8-9). Unter kontinuierlichen Daten werden alle Skalenniveaus gefasst, für die eine beliebige Anzahl an Zwischenausprägungen besteht: beispielsweise eine Zeitdauer. Umgekehrt haben diskrete Daten eine begrenzte Anzahl an Ausprägungen, die nicht weiter abgestuft werden können. Es ist zum Beispiel nicht möglich, dass eine Familie 1.7 Kinder hat. Das Skalenniveau definiert die Art der Ausprägung (Werte) der gemessenen Dimensionen (Merkmale), die in einer Variable erfasst werden. Bei quantitativen Forschungsmethoden werden die Skalen sehr genau definiert, da das Skalenniveau die rechnerischen Operationen und die Vergleichsmöglichkeiten definiert (Diaz-Bone 2006, Kapitel 2.1, 2.2; A. Field, Miles und Z. Field 2012, Kapitel 1.5.1). Für die Anwendung der Aesthetics ist aber die Unterscheidung zwischen kontinuierlichen und diskreten Skalen entscheidend.

Variablentyp	Skala	Beispiel	Beschreibung
Quantitativ/ numerisch	Kontinuierlich	1,3; 83; 1.5 x 10 ⁻²	Beliebige numerische Werte. Diese können ganze, rationale oder reelle Zahlen sein.

Variablentyp	Skala	Beispiel	Beschreibung
Quantitativ/ numerisch	Diskret	1, 2, 3, 4	Zahlen in diskreten Einheiten sind meistens ganze Zahlen. Ausnahmen: Bspw. die Werte 0.5, 1.0, 1.5 sind auch diskrete Werte, sofern im Datensatz keine dazwischen liegenden Werte existieren können .
Qualitativ/ nominal	Diskret	Hund, Katze, Fisch	Eindeutige Kategorien ohne feste Reihenfolge. Oft als Merkmale bezeichnet.
Qualitativ/ ordinal	Diskret	schlecht, angemessen, gut	Eindeutige Kategorien mit fester Reihenfolge. Oft als geordnete Merkmale bezeichnet.
Datum oder Zeit	Kontinuierlich oder diskret	5. Jan 2018, 08:03h	Spezifische Tage und/oder Zeiten. Allgemeine Datumsangaben ohne Jahr sind auch möglich ("4. Juli"). Das Format kann variieren.
Text	Keine oder diskret	Franz jagt im Taxi quer durch Bayern.	Freitext. Kann bei Bedarf als kategorisierbar behandelt werden.

Tabelle 2.2: Variablentypen für Visualisierungen. Quelle: Wilke (2020), S. 9. Eigene Anpassungen.

Die Tabelle 2.3 zeigt die verschiedenen Variablentypen. Die Spalte *Jahr* ist ein diskreter Datumwert, da kein anderes Jahr in der Variable vorkommt. *Gemeinde* und *Abfallart* sind beides nominale (kategorische) Werte ohne logische Reihenfolge. *Menge in Tonnen* ist ein kontinuierlicher numerischer Wert.

Jahr	Gemeinde	Abfallart	Menge in Tonnen
2021	Aeugst a.A.	Brennbare Abfälle und Sperrgut	323
2021	Affoltern a.A.	Brennbare Abfälle und Sperrgut	2188
2021	Bonstetten	Brennbare Abfälle und Sperrgut	871
2021	Hausen a.A.	Brennbare Abfälle und Sperrgut	603

Tabelle 2.3: Abfallart und -menge im Kanton Zürich. Datenquelle: Teil des biviz Pakets.

Für die Abbildung der Daten auf den Aesthetics wird angegeben, welche Datenwerte welchem Wert auf der Datenskala entsprechen. Das bedeutet, mithilfe der Skala erfolgt die eindeutige Zuordnung zwischen Daten und Aesthetics. Bei einem Diagramm mit einer x-Achse wird angegeben, welcher Wert auf welcher Position auf dieser Achse dargestellt

wird. Der gleichen Logik folgend, wird angegeben welche Grösse, Farbe oder Form ein Datenwert einnehmen soll. Anstatt eine Positionsskala wird eine Grössen-, Farben- oder Formskala verwendet. Ein Datenwert entspricht in jeder Skala einem eindeutigen Skalawert bzw. Aesthetic. Je Datenwert darf es nur einen Skalawert geben und umgekehrt. Eine eins zu eins Beziehung ist notwendig, damit die Datenvisualisierung nicht mehrdeutig wird (vgl. Kapitel 3.2).

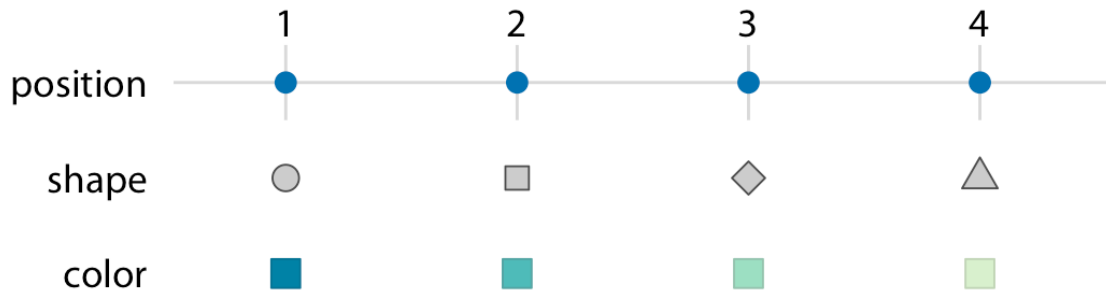


Abbildung 2.1: Skalen verknüpfen Datenwerte mit Aesthetics. Quelle: Wilke (2020), S. 10.

Wenden wir diese Erkenntnisse beim Datensatz zum Abfall im Kanton Zürich an, dann erhalten wir folgendes Ergebnis: Auf der x-Achse werden die *Gemeinden* anhand der Positionsskala platziert. Die Variable hat keine logische Reihenfolge. Für die Unterstützung des Lesens des Diagramms (Abbildung 2.2), sind die Gemeinden anhand der Menge an Abfall sortiert. Auf der Positionsskala y-Achse ist die *Menge in Tonnen* übertragen. Zusätzlich wurden die *Gemeinden* auf die Farbskala übertragen. Vor dem Hintergrund des Data Storytelling lässt sich die Frage stellen, ob die Verwendung des Aesthetic Farbe sinnvoll ist (vgl. Kapitel 2.2). Bei diesem Beispiel steht jedoch die Beziehung zwischen den Datenwerten und Aesthetics durch Skalen im Zentrum. Daher wurde für jede Gemeinde eine eigene Farbe verwendet.

2.1 Das kartesische Koordinatensystem als Positionsskala

Bei der Abbildung 2.2 entspricht die Positionsskala einem kartesischen 2D-Koordinatensystem. Jeder Ort ist durch einen x- und y-Wert eindeutig markiert. Da die Achsen positive als auch negative Zahlen darstellen, muss der Zahlenbereich für jede Achse definiert werden. Bei diesem Beispiel verläuft die y-Achse von 0 bis 2297.4 (5% über dem höchsten y-Wert, welcher bei diesem Beispiel 2188 ist). Ist ein Datenwert innerhalb des definierten Zahlenbereichs, dann wird er im Diagramm an der entsprechenden Position abgebildet. Ansonsten wird der Wert verworfen und erscheint nicht im Diagramm (Wilke 2020, S. 13).

Beim kartesischen Koordinatensystem sind die Abstände zwischen den Gitterlinien der Achsen diskrete Schritte. Entlang einer Achse sind die Gitterlinien gleichmässig verteilt und entsprechen einer linearen Positionsskala. Dies gilt sowohl für die Dateneinheiten als auch in der Visualisierung. In allen drei Diagrammen der Abbildung 2.3 sind die Schritte auf der y-Achse je Gitterlinie 3000 und auf der x-Achse 10 Jahre. Der verwendete Raum um die diskreten Schritte abzubilden können aber unterschiedlich gross sein, womit

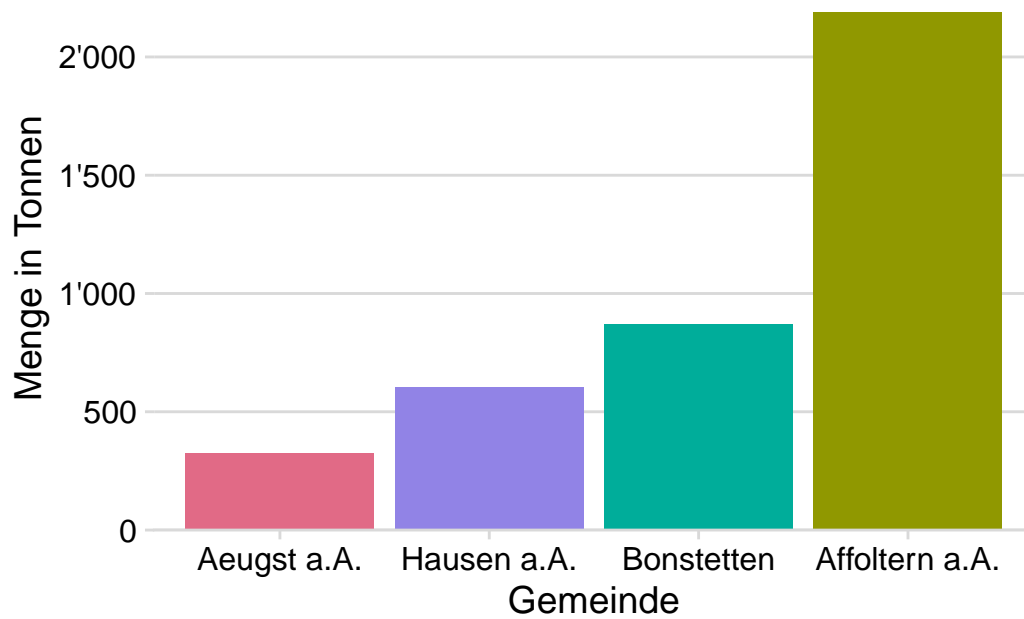


Abbildung 2.2: Anzahl brennbare Abfälle und Sperrgut je Gemeinde im Jahr 2021. Eigene Darstellung.

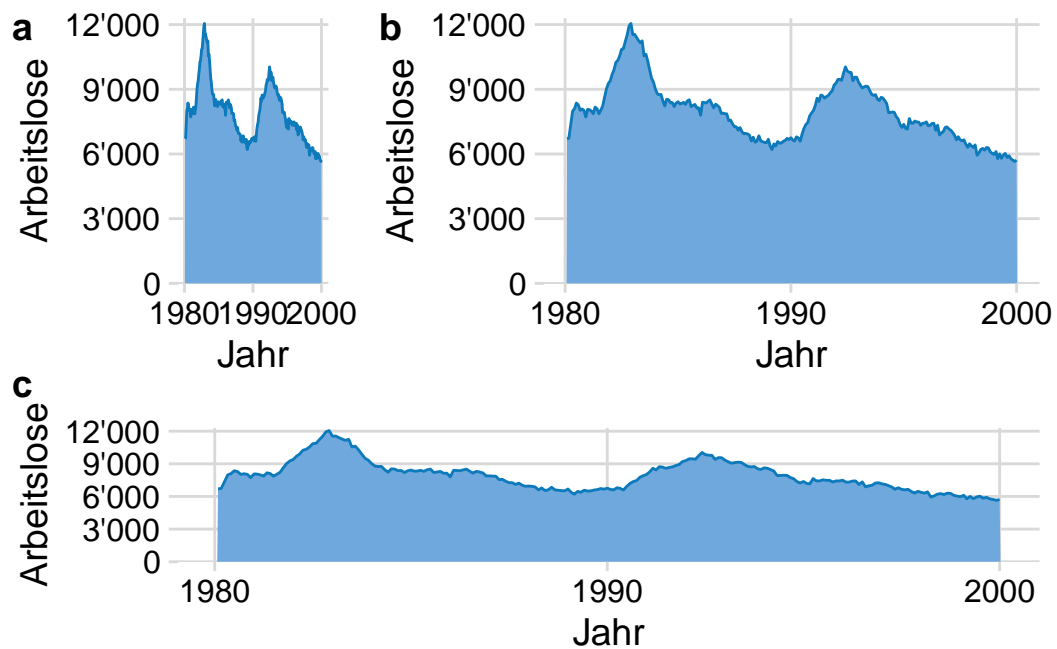


Abbildung 2.3: Die Anzahl Arbeitslose sind in Tausend angegeben. Datenquelle: Teil des ggplot2 Pakets. Eigene Darstellung.

unterschiedliche Botschaften vermittelt werden. Abbildung (a) betont die Veränderung auf der y-Achse und Abbildung (c) betont die Veränderung über die Zeit. Werden auf der x- und y-Achse die gleichen Einheiten verwendet, so sollten die Gitterabstände identisch sein. So, dass der Abstand zwischen zwei Gitterlinien die gleiche Menge an Dateneinheiten beinhaltet (Wilke 2020, S. 14-15).

Neben linearen Achsen gibt es auch nichtlineare Skalen, welche meistens eine logarithmische Skala verwendet. Hier entspricht eine Einheit auf der Skala einer Multiplikation von einem festen Wert (Wilke 2020, S. 17). Bei Datenvisualisierungen werden auch Polarkoordinatensysteme verwendet, bei denen die Position durch den Winkel und radialen Abstand zum Ursprung angegeben wird (Wilke 2020, S. 22). Da bei dieser Arbeit, ausser beim Donutdiagramm, nur das lineare kartesische Koordinatensystem verwendet wird, stehen diese Systeme nicht im Fokus. Für eine Vertiefung der Thematik ist das *Kapitel 3* aus Wilke (2020) zu empfehlen.

2.2 Farbe als Steuerungsinstrument

Bei Datenvisualisierungen sind Farben ein wirkungsvolles Mittel, um die Betrachtenden beim Lesen des Diagramms zu unterstützen und ihre Aufmerksamkeit zu lenken. Damit die Wirkung sich entfalten kann, muss die Anwendung von Farben selektiv und überlegt sein. Die Verwendung von Farbe sollte ein bewusster Entscheid mit einer strategischen Absicht sein. Welche Aspekte sollen die Aufmerksamkeit der Lesenden erhalten? Damit die Farben wirken, benötigt es einen Kontrast. Die Gitterlinien der bisherigen Grafiken sind alle in grau. Farben heben sich besser von grau als von blau ab, dadurch entsteht ein grösserer Kontrast zu den Farben, die für die Lenkung der Lesenden verwendet werden. Für die grösste Wirkung der Farben werden sie sparsam und konsistent eingesetzt. So wird gewährleistet, dass sie ihre präattentive Wirkung beibehalten. Ist alles unterschiedlich, dann kann nichts hervorstechen (Nussbaumer Knaflitz 2017, S. 98-99). Unter präattentive Merkmale versteht Nussbaumer Knaflitz Kennzeichnungen, die eine vorbewusste Wahrnehmung von Sinnesreizen stimulieren und effizient mit dem ikonischen Gedächtnis interagieren. Das ikonische Gedächtnis ist aktiviert, sobald wir etwas betrachten. Dabei nehmen wir seine Tätigkeit nicht bewusst wahr. Bereits nach einem Sekundenbruchteil wird das Signal an das Kurzzeitgedächtnis weitergeleitet. Hier werden die Informationen verarbeitet. Da das Kurzzeitgedächtnis nur eine begrenzte Kapazität hat, müssen beim Data Storytelling die Reize für das Gehirn bewusst gesteuert werden. Die kognitive Belastung für das Publikum soll möglichst gering sein, damit es die vermittelten Informationen aufnehmen kann. Das Kurzzeitgedächtnis kann in etwa vier Elemente visueller Informationen zeitgleich verarbeiten. Indem Informationen als zusammenhängende visuelle Elemente dargestellt werden, wird die kognitive Belastung für das Publikum reduziert und besitzt dennoch eine hohe Informationsdichte. Präattentive Merkmale helfen die Aufmerksamkeit des Publikums zu steuern und eine visuelle Hierarchie in einem Diagramm zu schaffen. Neben Farben sind auch Formen, Grösse oder Positionen klassische präattentive Merkmale (Nussbaumer Knaflitz 2017, S. 83-86). Beispielsweise können wir dank der präattentiven Funktion der Farbe einfacher und schneller die Anzahl der Zahl drei in einem Zahlenblock zählen Abbildung 2.4a und Abbildung 2.4b.

756395068473	756 3 95068473
658663037576	65866 3 0 3 7576
860372658602	860 3 72658602
846589107830	8465891078 3 0

(a) Ohne präattentivem Merkmal

(b) Mit präattentivem Merkmal

Abbildung 2.4: Zahlen mit und ohne präattentivem Merkmal. Quelle: (Nussbaumer Knaflic 2017, S. 86).

Die korrekte Verwendung von Farbpaletten und ihre Wichtigkeit für die Kommunikation mit Datenvisualisierungen betrifft neben der Interpretation der Diagrammen auch die Berücksichtigung der Farbenblindheit (Hawkins 2015; Bartram, Patra und Stone 2017). Die wichtigsten Aufgaben von Farben ist die Unterscheidung von Datengruppen (wie bei Abbildung 2.4b), das Darstellen von Datenwerten oder die Hervorhebung von Datenpunkten. Bei Abbildung 2.2 dient die Farbe als Unterscheidungsmerkmal der einzelnen Gemeinden. Bei der Anwendung von Farbe als Unterscheidungsmerkmal werden qualitative Farbskalen verwendet. Das heisst, die Anzahl Farben ist endlich, sie unterscheiden sich voneinander und sind gleichwertig zueinander. Folglich darf keine Farbe dominanter als die andere sein und der Eindruck einer Reihenfolge muss vermieden werden (Wilke 2020, S. 25-26). Die Okabe Ito Farbskala (Okabe und Ito 2008) ist eine bekannte Standardskala, welche die beschriebenen Voraussetzungen erfüllt und Farbenblindheit berücksichtigt. Rund acht Prozent der Männer und ein halbes Prozent der Frauen sind farbenblind, wodurch sie Rottöne und Grüntöne nur schlecht unterscheiden können.¹ Um positive und negative Punkte trotzdem mit Farben hervorzuheben, wird oft blau für positive Werte und orange für negative Werte verwendet (Nussbaumer Knaflic 2017, S.101-102). Um quantitative Datenwerte darzustellen werden sequenzielle Farbskalen verwendet, bei denen erkennbar ist, welcher Wert grösser oder kleiner ist. Damit abgeschätzt werden kann, wie weit zwei Werte voneinander entfernt sind, müssen sich die Farbabstufungen gleichmässig über den gesamten Bereich verändern. Das ist sowohl mit einem einzelnen Farbton, als auch mit mehreren Farbtönen möglich. Werden quantitative Datenwerte relativ zu einem neutralen Mittelpunkt visualisiert, wird eine divergente Farbskala verwendet. Beispielsweise bei einer Variable mit positiven als auch negativen Werten (Wilke 2020, S. 27-28).

Wann ist der Einsatz von Farbe sinnvoll? Immer dann, wenn sie dem Publikum das Lesen des Diagramms erleichtert. Farbe muss nicht unterhalten, sondern ein Signal senden. Verändert sich etwas in der Grafik oder gibt es einen wichtigen Punkt zu beachten, so muss das Publikum auf diesen Umstand aufmerksam gemacht werden. Das geht nur, wenn die Farben zurückhaltend und durchgehend sind, also nicht in jeder Visualisierung eine andere

¹Es gibt viele Programme um die Farbenblindheit zu simulieren. Beispielsweise die Webseiten hclwizard.org/cvdemulator oder vischeck.com.

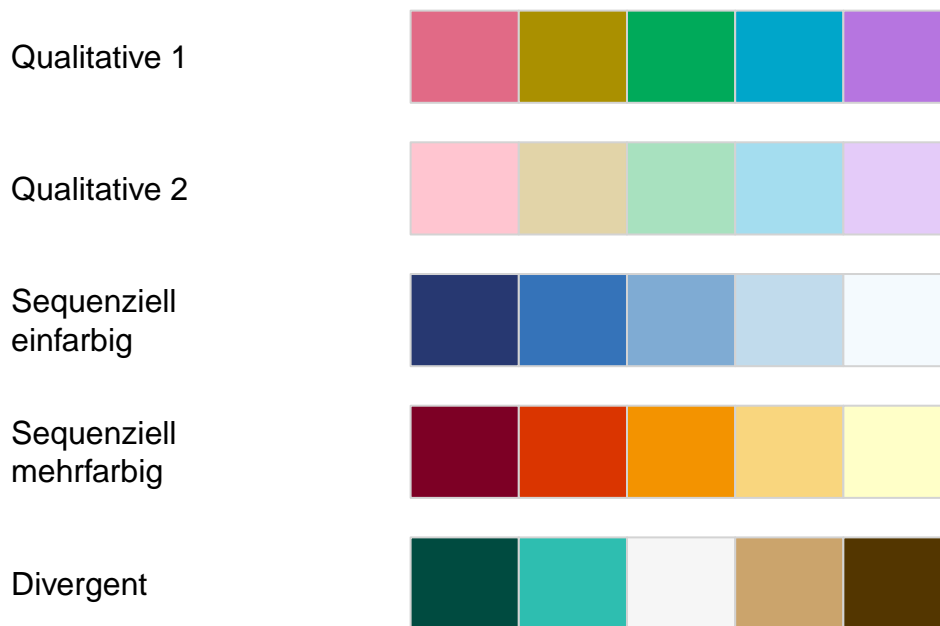


Abbildung 2.5: Das R Paket colorspace (Zeileis u. a. [2020](#)) ist ein flexibles Werkzeug um eigene Farbpaletten zu erstellen (beispielsweise eine Okabe Ito Farbskala) oder auf bestehende Skalen zuzugreifen. Eigene Darstellung.

Farbe verwendet wird. Die kognitive Belastung reduziert sich, wenn die Bedeutung einer Farbe innerhalb eines Visualisierungsportfolio (beispielsweise in einem Paper oder in einer Präsentation) gleich bleibt (Nussbaumer Knaflic [2017](#), S. 100-101). Ansonsten besteht nicht nur die Gefahr, dass das Publikum verwirrt, sondern gar aktiv in die Irre geführt wird (Borland und Taylor Ii [2007](#), S. 15).

2.3 Gestaltprinzipien in Datenvisualisierungen

Die Aufnahme von Informationen ist für das Gehirn eine mentale Anstrengung. Die Gehirnkapazität des Publikums ist begrenzt, daher muss die kognitive Belastung bewusst und effektiv gestaltet sein. Das bedeutet, relevante Informationen müssen mit möglichst geringer wahrgenommener kognitiver Belastung kommuniziert werden. Dafür müssen die Signale (die Informationen, die vermittelt werden) gestärkt und das Rauschen (Elemente die von der Information ablenken) reduziert werden. Das Diagramm soll eingänglich und einfach erscheinen. Komplizierte Grafiken können das Publikum abschrecken und ihre Aufmerksamkeit geht verloren. Die Gestaltprinzipien der visuellen Wahrnehmung helfen die Signale einer Grafik zu erkennen und das Rauschen zu minimieren, damit die Botschaft der präsentierten Daten einfach erkennbar ist (Nussbaumer Knaflic [2017](#), S. 61-63). Die Überlegungen und Grafiken zu den Gestaltprinzipien folgen den Ausführungen von Nussbaumer Knaflic ([2017](#)) im Kapitel *Gestaltprinzipien der visuellen Wahrnehmung*:



Abbildung 2.6: Das Gestaltprinzip der Nähe.

Durch die physische Nähe der Punkte, werden sie als zusammengehörig wahrgenommen. Dank der Anordnung und Nutzung von Zwischenräumen sehen wir Gruppen, Linien etc.. Bei den vertikalen Reihen ist der Weissraum (Zwischenräume) zwischen den nebeneinander liegenden Punkten grösser und bei den horizontalen Reihen zwischen den untereinander liegenden Punkten.

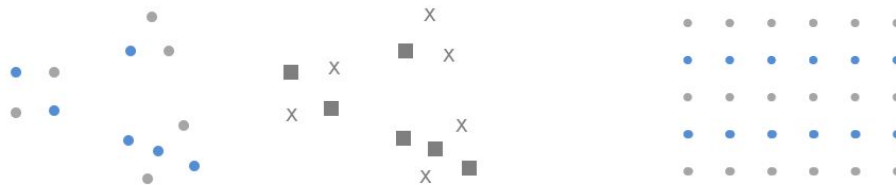


Abbildung 2.7: Das Gestaltprinzip der Ähnlichkeit.

Haben Objekte eine ähnliche Erscheinung (Form, Farbe, Grösse, etc) so stellt unser Gehirn eine Verbindung zwischen diesen Punkten her. Damit kann dem Publikum wichtige Interpretationshilfe geboten werden und die kognitive Belastung ist tief. Beispielsweise sehen wir durch die Färbung der Punkte in der rechten Grafik Linien.

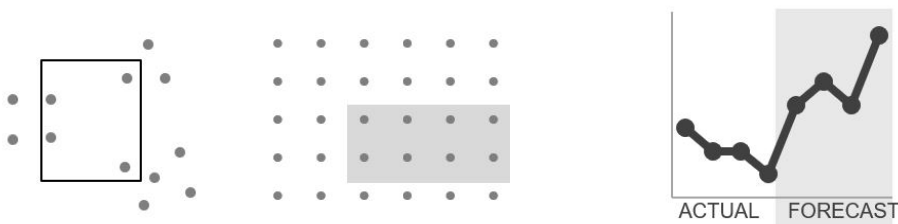


Abbildung 2.8: Das Gestaltprinzip der Umrandung.

Durch Umrandungen werden Teilmengen einfach als Gruppe identifiziert. Die Fläche kann schattiert oder klassisch mit Linien umrandet werden. Die Umrandung ist hilfreich zur Unterscheidung von unterschiedlichen (Daten-) Bereichen.

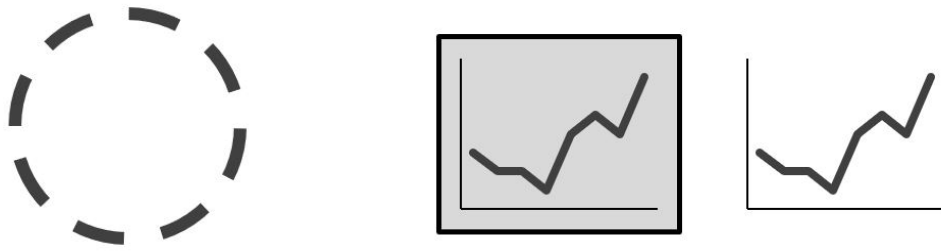


Abbildung 2.9: Das Gestaltprinzip der Form.

Unser Gehirn neigt dazu, einzelne Elemente in einer Reihe wenn möglich als eine einzige Form wahrzunehmen. Trotz Lücken in der Kreisform, erkennt unser Gehirn einen Kreis, indem er die fehlenden Teile ausfüllt. Dazu muss das Konstrukt aber bekannt und einfach dargestellt sein. Dank dem Konzept der Form erkennen wir zudem, die Zusammengehörigkeit einer Grafik ohne einen Rahmen. Gleichzeitig wird das Rauschen reduziert und die Daten hervorgehoben.

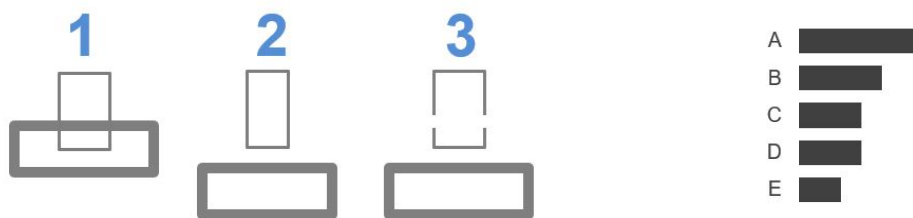


Abbildung 2.10: Das Gestaltprinzip der Kontinuität.

Unser Gehirn sucht in Objekten die Kontinuität und das Bekannte. Daher erwarten die meisten Menschen beim Auseinander nehmen des Bildes 1, das Bild 2. Es könnte aber genauso gut Bild 3 sein (vgl. auch Kanizsa [1970](#)). Im Balkendiagramm wird keine y-Achse benötigt, da unser Hirn die Linie selber zeichnet und erkennt, dass alle Balken am gleichen Punkt starten. Da die Distanz bzw. der Weissraum zwischen der Beschriftung (links) und den dargestellten Daten (rechts) überall identisch ist, wird eine Linie signalisiert.

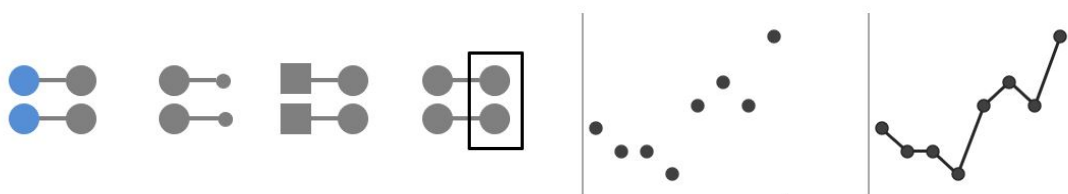


Abbildung 2.11: Das Gestaltprinzip der Verbindung.

Das Gestaltprinzip der Verbindung ist stärker als das Prinzip der Ähnlichkeit und schwächer als die Umrandung. Das heisst, die Zusammengehörigkeit von verbundenen

Elemente ist stärker als die gleiche Farbe, Grösse oder Form. Eine Umrandung besitzt jedoch eine höhere visuelle Hierarchie als die Verbindung, wodurch die Objekte in einem Rahmen zusammengehörig wahrgenommen werden obwohl sie mit anderen Objekten verbunden sind. Verbindungen helfen, den Datenpunkten eine Ordnung zu geben und werden unter anderem bei Liniendiagrammen eingesetzt. Die meisten Liniendiagramme bestehen aus einzelnen Datenpunkten, welche erst dank dem Gestaltprinzip der Verbindung als Kontinuum erkennbar sind.

2.4 Mit Daten Geschichten erzählen

Die Vermittlung von Analysen ist üblicherweise der einzige Teil, welcher das Publikum zu sehen bekommt von einer Datenanalyse. Folglich sollte dieser Schritt möglichst erfolgreich und nachhaltig gestaltet sein. Damit das erreicht wird, muss der Wechsel von der erforschenden Analyse zur erklärenden Analyse erfolgen. Die erforschende Analyse ist der Prozess, welcher zu neuem Verständnis und neuen Erkenntnissen führt. Bei der Vermittlung der Resultate kann aber nicht der gesamte analytische Prozess wiedergegeben werden, daher ist es wichtig einen erklärenden Ansatz anzuwenden. Hier wird eine bestimmte Erkenntnis erklärt und analysiert, die für das Publikum und die gegebene Situation wichtig ist (Nussbaumer Knaflitz 2017, S. 17).

	erforschende Analyse	erklärende Analyse
Ziel	Verstehen	Kommunizieren
Publikum	Analyst:in (sich selbst)	andere Personen
Vertrautheit des Publikums mit den Daten	Sehr vertraut	Nicht bis wenige vertraut
Fokus der Visualisierung	Flexibel und schnell	Einfach, klar und schlüssig
Erzählung	Unbekannt	Bekannt
Resultat	Insight (Verständnis & Erkenntnis)	Aktion

Tabelle 2.4: Die zwei Arten der Analyse. Quelle: Dykes (2020), S. 138. Eigene Anpassungen.

In der Forschung ist die erklärende Position die Beantwortung der Forschungsfrage und im geschäftlichen Umfeld beispielsweise die Erklärung, weshalb der Pendenzenstand im Team A zugenommen hat. Um die Fragen zu beantworten, werden eine Vielzahl von erforschenden Analysen durchgeführt, aber nur einige wenige geben Informationen, um die Fragestellung zu beantworten. Auf diese Informationen wird bei der erklärenden Analyse eingegangen. Damit die Argumente überzeugen und die Erkenntnisse in Erinnerung

bleiben, wird eine packende Geschichte benötigt. Die Erzählung einer Geschichte hilft, die gewonnenen Fakten aus der erforschenden Analyse bei der erklärenden Analyse dem Publikum interessant und nachvollziehbar zu präsentieren. Damit das möglich ist, muss der Kontext eruiert werden. Wer ist das Publikum und in welcher Rolle stehe ich zum Zielpublikum? Je nach Situation muss eine andere Kommunikation gewählt werden, damit die Geschichte ihr Ziel erreicht. Was soll das Publikum wissen? Erkennen was für das Publikum spannend und relevant ist. Hier wird die Handlung der Geschichte definiert. Dabei muss immer die Frage beantwortet werden, weshalb soll sich das Publikum für die Geschichte interessieren. Wie kann die Frage beantwortet werden? Bei diesem Schritt werden die Daten ausgewählt, welche die Geschichte untermauern und plausibilisieren (Nussbaumer Knaflitz 2017, S. 18-23).

Was aber ist eine Geschichte? Bei einer Geschichte werden Beobachtungen, Erkenntnisse und Ereignisse in eine bestimmte Reihenfolge gebracht, mit dem Ziel beim Publikum eine emotionale Wirkung zu erzielen. Durch den Aufbau einer Spannung am Anfang und der Lösung am Ende der Geschichte wird das Publikum gefesselt, was eine emotionale Reaktion auslöst (Wilke 2020, S. 304). Neben der Wiederholung ist die emotionale Wirkung der Geschichte durch den Spannungsbogen wichtig, damit sie in unserem Gedächtnis bleibt. Es gibt eine Vielzahl von narrativen Strukturen und Methoden des Storytellings, sei es im Theater, im Film oder bei Texten. Eine einfache Methode für das Data Storytelling ist das Muster *Anfang-Mitte-Ende*, welches dem klassischen Narrativ eines Forschungspapers (Einleitung-Hauptteil-Schlussfolgerung) gleicht. Am Anfang wird die Handlung eingeführt und eine Auslegeordnung gemacht, damit das Publikum erkennt, weshalb die Geschichte wichtig für sie ist. Was ist der Kontext, welches sind die Schlüsselpunkte, was ist das Problem und was ist die gewünschte Lösung? In der Mitte muss das Publikum von einem Sachstand überzeugt werden. Deshalb steht das *wie* der Problemlösung im Zentrum. Dafür wird das Thema in seinen Details (Hintergrundinformationen, Vergleiche, Szenarien, mögliche Problemlösungen) dargestellt. Am Ende wird auf die Ausgangslage Bezug genommen und eine Handlungsempfehlung abgegeben. Über den ganzen Prozess hinweg steht immer das Publikum im Mittelpunkt: Warum ist es relevant für sie? Was stößt auf Resonanz und motiviert? Dabei steht jedes Mal der gleiche Inhalt zur Debatte, lediglich aus unterschiedlichen Perspektiven (Einführen, Details, Zusammenfassen) (Nussbaumer Knaflitz 2017, S. 146-149).

Bei der Integration von Datenvisualisierungen in eine Geschichte, müssen gewisse Punkte berücksichtigt werden. Erstens kann eine einzelne Visualisierung keine ganze Geschichte erzählen. Damit mit Visualisierung eine Geschichte erzählt werden kann, benötigt es mehrere Diagramme. Pro Station des Storytellings braucht es in der Regel mindestens eine Grafik. Der Anfang, die Mitte und das Ende einer Geschichte können nicht in einem Satz sinnvoll dargelegt werden. Das gilt auch für Visualisierungen. Eine einzelne Visualisierung kann keine Geschichte erzählen (Wilke 2020, S. 305). Damit die Visualisierung (Form) den gewünschten Effekt (Funktion) auslöst, muss die Form (Visualisierung) der Funktion (Effekt) folgen. Das heißt, die Visualisierung soll etwas auslösen, damit das funktioniert, benötigt sie eine vorher definierte Funktion (Was soll die Grafik machen?). Um die Funktion des Diagramms aufzuzeigen, muss dem Publikum aufgeführt werden, wie es mit der Darstellung umgehen soll. Die wichtigste Methode dafür ist bei der Datenvisualisierung die Verwendung von Farben (vgl. Kapitel 2.2). Damit die Hervorhebung seine Wirkung entfalten kann, ist es wichtig sparsam mit diesem Tool umzugehen (Nussbaumer Knaflitz 2017, S. 107-109). Zudem muss die Visualisierung zugänglich sein. Nicht das Publikum

ist verantwortlich, dass es die Grafik versteht, sondern die Autorschaft. Das wird erreicht durch ein verständliches Design (keine unnötige Komplexität), Lesbarkeit (Schriftart und Grösse), Klarheit (wichtiges wird hervorgehoben) und einfache Sprache (an Publikum angepasst). Das Publikum muss durch das Diagramm geführt werden, dazu sind Titel, Achsenbeschriftungen und direkte Annotationen notwendig (Nussbaumer Knaflitz [2017](#), S. 118-119). Werden alle diese Punkte berücksichtigt und alle Elemente folgen einer visuellen Ordnung, dann hat das Diagramm eine ansprechende Ästhetik, wodurch das Vertrauen des Publikums gewonnen wird (Nussbaumer Knaflitz [2017](#), S. 68, S. 123).

3 Vorgehen und Methode

3.1 Tidy Data als Basis von Datenvisualisierungen

Für die effiziente Anwendung von Datenvisualisierungen müssen die Daten als *Tidy Data* (Wickham 2014) aufbereitet sein. Folgende Kriterien müssen gemäss Wickham (2014, S. 4) erfüllt sein, damit ein Datensatz dem Grundsatz Tidy Data entspricht:

1. Jede Variable hat eine eigene Spalte.
2. Jede Beobachtung hat eine eigene Zeile.
3. Jeder Wert hat eine eigene Zelle.

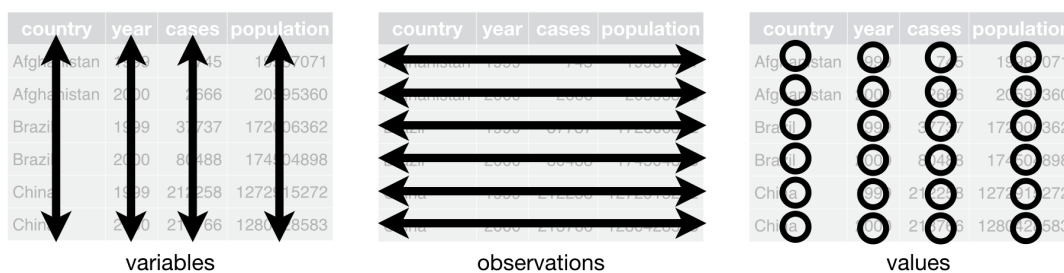
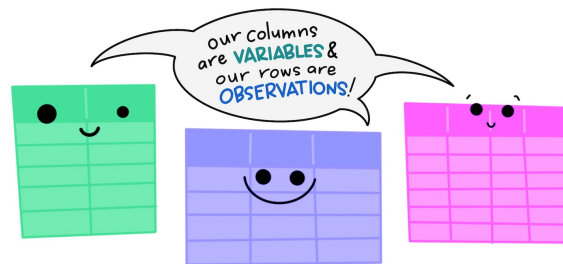


Abbildung 3.1: Bei einem tidy Datenset sind Variablen in Spalten, Beobachtungen in Zeilen und Werte in Zellen gespeichert. Quelle: Wickham und Grolemund (2016), S. 149.

Die Standardisierung der Organisation von Datenwerten in einem Datensatz vereinfacht die Datenanalyse. Die Datenstruktur (Zweidimensionale Tabelle mit Spalten und Zeilen) wird beim Tidy Data Ansatz mit einer Semantik (Variablen, Beobachtungen, Werte) verknüpft. Es ist nicht immer einfach herauszufinden, was Beobachtungen und was Variablen sind. Um die Variablen und Beobachtungen für eine Datenanalyse herauszufinden, gilt der Grundsatz: Beziehungen lassen sich einfacher zwischen Variablen beschreiben und Vergleiche zwischen Gruppen von Beobachtungen (Wickham 2014, S. 3-4).

Abgesehen davon, dass mit Tidy Data eine konsistente Methode für die Datenspeicherung verwendet wird, kann R die Vorteile einer vektorisierten Programmiersprache so optimal ausschöpfen (Wickham und Grolemund 2016, S. 150). Vektorisiert bedeutet, dass eine Funktion einen Vektor mit Werten als Input nimmt und einen Vektor mit der gleichen Anzahl an Werten als Output retourniert (Wickham und Grolemund 2016, S. 56). Tidy

The standard structure of tidy data means that "tidy datasets are all alike..."



"...but every messy dataset is messy in its own way."

—HADLEY WICKHAM

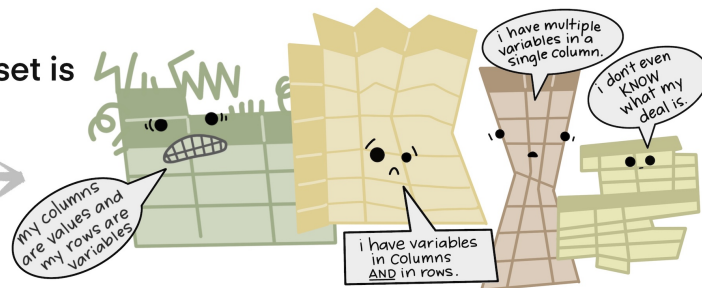


Abbildung 3.2: Ordnung durch Tidy Data. Quelle: Lowndes und Horst (2020).

Data entspricht einem langen Datenformat. Dadurch entsteht eine Redundanz an Werten, was bei der Speicherung von Daten nicht gewünscht ist, aber die Datenanalyse vereinfacht (Healy 2018, S. 56).

3.2 Die Beziehung von Daten und Visualisierungen mit Grammar of Graphics

Die Datenvisualisierungen im entwickelten Paket `{biviz}` basieren auf dem Paket `{ggplot2}` (Wickham 2016), welches den Ansatz *Grammar of Graphics* (Wilkinson 2005) verfolgt und vektorisiert ist. Durch die Grammar of Graphics werden Beziehungen zwischen den Daten und ihrer grafischen Darstellung ausgedrückt und bietet eine einheitliche Möglichkeit, Visualisierungen zu erstellen. In der Theorie folgt die Grammar of Graphics sieben Schritten zur Erstellung einer Grafik: Daten werden Variablen zugeordnet. Anschliessend durchlaufen die Variablen drei mögliche Transformationsschritte (Algebra, Skalen, Statistik). Danach werden die transformierten Variablen einem geometrischen Objekt übergeben, damit die Daten eine Form erhalten. Im nächsten Schritt wird das Objekt in einem Koordinatensystem positioniert. Zum Schluss wird ein visuell wahrnehmbares Objekt erstellt, das Grafik heisst (Wilkinson 2012, S. 376-377). Diese strukturierte Beziehung zwischen den Datenvariablen und deren Repräsentation in einer Grafik macht sich das Paket `{ggplot2}` zu eigen um Diagramme zu erstellen. Dazu werden zuerst die Daten definiert, anschliessend wird das visuelle Element gewählt und am Ende werden die Details einer Visualisierung angepasst. Das heisst, ein Code mit `{ggplot2}` folgt bei der Erstellung einer Grafik einer logischen Struktur. Es werden Verbindungen zwischen den Datenvariablen und den Skalen der grafischen Elementen einer Visualisierung (Farbe, Form, Position, Grösse) hergestellt. Diese Verbindungen

heissen *aesthetics* (vgl. Kapitel 2). Am Ende des Datenvisualisierungsprozesses wird diese Verbindung in eine Grafik umgewandelt. Damit eine solche Verbindung entstehen kann, werden in der Funktion `ggplot()` die Daten sowie die Beziehung zwischen den Datenvariablen und dem *mapping* auf die aesthetics definiert. Anschliessend wird dem Programm gesagt, welcher Diagrammtyp (Balkendiagramm, Streudiagramm, etc.) dargestellt werden soll. Diagrammtypen werden anhand der *geom* definiert. Mit dem *geom* wird gesagt, welches geometrische Objekt (Balken, Punkte, etc.) für das Diagramm benutzt wird. Das heisst, Balkendiagramme werden mit `geom_bar()` oder Streudiagramme mit `geom_point()` erzeugt. Diese Abfolge von Code reicht, damit `{ggplot2}` eine Grafik erstellen kann. Um mehr Details der Grafik, wie Achsen, Skalen oder Beschriftungen zu kontrollieren, werden weitere Codestücke hinzugefügt. Dafür wird die gleiche Logik wie beim Definieren der Diagrammtypen benutzt (Healy 2018, S. 54-56). Viele Schritte der Grammar of Graphics haben bei `{ggplot2}` abhängig von den Daten und dem gewählten *geom* eine Voreinstellung, beispielsweise bei den Transformationsschritten und dem Koordinatensystem. Die Einstellungen können aber manuell angepasst werden. Jeder Teilaspekt hat eine eigene Funktion, welche Argumente besitzt, die spezifizieren was an der Grafik angepasst wird.

Durch die Zerstückelung der einzelnen Schritte der Erstellung einer Datenvisualisierung, kann systematisch Stück für Stück eine individuelle Grafik erstellt werden. Dabei basieren die unterschiedlichen Grafiktypen immer auf den oben beschriebenen Grundspezifikationen. Am Ende besteht ein Diagramm immer aus einer Kombination von vier Quellen: 1. Daten, dessen Datenwerte dargestellt werden. 2. Skalen und Koordinatensystem, um die Daten in einer Grafik abzubilden. 3. Eine geometrische Form zur Definition des Erscheinungsbildes der Grafik. 4. Erläuterungen wie Labels oder Titel, damit die Grafik interpretierbar wird (Wickham 2010, S. 4-5).

Die Umsetzung in `{ggplot2}` folgt immer dem gleichen Konzept (Healy 2018, S. 60):

1. Daten als Tidy Data aufbereiten.

```
ggplot2::mpg |>
  dplyr::select(manufacturer:cyl, class) |>
  head() |>
  knitr::kable()
```

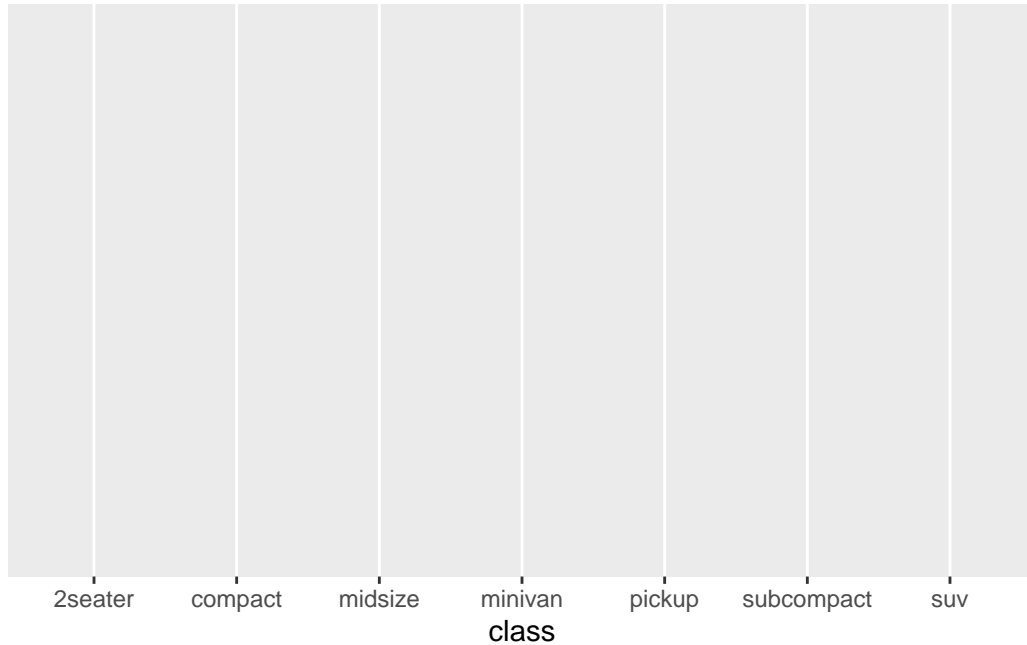
manufacturer	model	displ	year	cyl	class
audi	a4	1.8	1999	4	compact
audi	a4	1.8	1999	4	compact
audi	a4	2.0	2008	4	compact
audi	a4	2.0	2008	4	compact
audi	a4	2.8	1999	6	compact
audi	a4	2.8	1999	6	compact

2. Daten definieren und entscheiden, welche Beziehungen abgebildet werden sollen (mapping).

```
plot <-
  ggplot(data = mpg,
```

```
mapping = aes(x = class)) # aesthetics
```

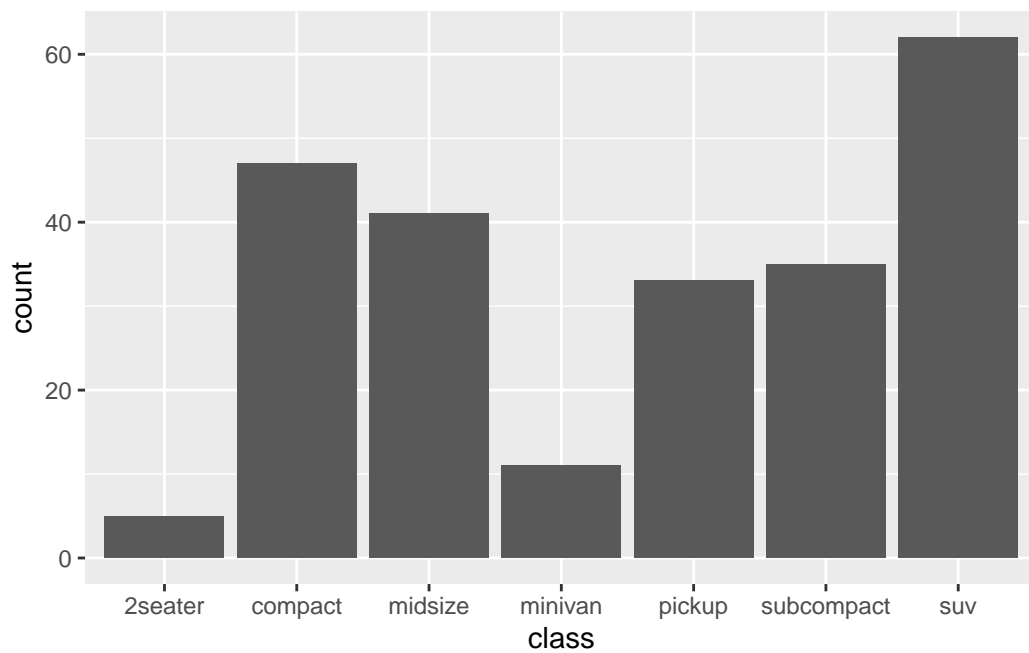
plot



3. Wahl des geometrischen Objekts. Es sind mehrere Layer möglich. Die aesthetics können je geom in der `geom_*()` Funktion angepasst werden.

```
plot <-  
plot +  
  geom_bar()
```

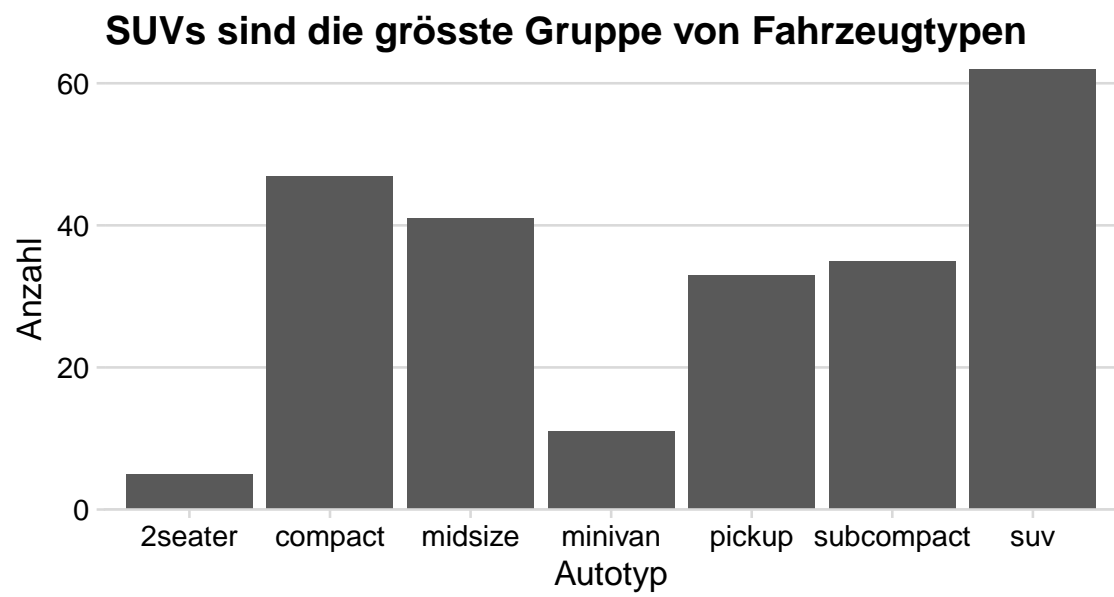
plot



4. Kontrolle von Details im Diagramm inkl. Koordinatensystem. Default ist das kartesische Koordinatensystem (Wilke 2020, S.13).

```
plot <-
  plot +
  scale_y_continuous(expand = c(0, NA)) +
  labs(
    title = "SUVs sind die grösste Gruppe von Fahrzeugtypen",
    x = "Autotyp",
    y = "Anzahl"
  ) +
  cowplot::theme_minimal_hgrid()

plot
```

4 R-Paket Entwicklung

4.1 Struktur im Prozess

Zu Beginn des Entwicklungsprozesses standen organisatorische Entscheide im Fokus. Zum einen musste die Ordnerstruktur des Pakets bestimmt und zum anderen eine Strategie für die Benennung der Funktionen definiert werden. Beide Aspekte helfen eine Struktur im Paket zu etablieren. Die Ordnerstruktur im `{biviz}` Paket folgt der Empfehlung von Bannert (Bannert 2022, Kapitel 3.2.5):

- R

Hier werden die Skripts der Funktionen gespeichert. Die einzelnen Funktionen des Pakets werden thematisch in einer Datei zusammengefasst. Dank dem Paket `{roxygen2}` (Wickham, Danenberg u. a. 2022a) kann die Dokumentation der Funktionen gleich im selben Skript wie die Funktionen geschrieben werden. Mit dem Paket `{devtools}` (Wickham, Hester u. a. 2022a) werden alle relevanten Dateien für die Dokumentation erzeugt. `{devtools}` erstellt diesen Ordner automatisch.

- man

In diesem Ordner wird im Normalfall die automatisch erzeugte Dokumentation abgespeichert und er beinhaltet die Funktion- und Datendokumentationen. Diese Dokumentationen werden bei der Anwendung des Pakets mithilfe `?function_name` oder `help(function_name)` aufgerufen. `{devtools}` erstellt diesen Ordner automatisch.

- docs

In diesem Ordner werden die Dokumente für das Bereitstellen einer GitHub Page abgelegt. Mit Hilfe von `{devtools}` werden die notwendigen Files erstellt und auf GitHub geladen. Die Website kann dabei in Markdown und R erstellt werden, wodurch keine grundlegenden Webentwicklungskills notwendig sind. Die erstellte GitHub Page greift auch auf die Dokumentationen im *man* Ordner zu und veröffentlicht diese unter dem Reiter *Verzeichnis*. Daher eignet sich die Website für die Dokumentation des Pakets. Nach dem Pushen von *docs* auf GitHub steht die Website unter `username.github.io/reponame` zur Verfügung. Bei `{biviz}` ist das: <https://tricktracktriu.github.io/biviz/>.

Wichtig: *docs* existiert im GitHub-Repository in einem separaten Zweig namens *gh-pages*. So ist die Website separiert von der Paketquelle (Wickham und Bryan 2022, Kapitel 20.3).

- data

Damit der Bezug der Daten relativ bleibt, werden sie in einem separaten Ordner abgespeichert. Der relative Bezug auf Daten und Funktionen (R) ist wichtig, damit andere Nutzende das Programm einfach verwenden können und der Pfad nicht aufgrund der persönlichen Ordnerstruktur gebrochen wird. Weitere Details zu relativen Datenpfaden hat Jenny Bryan in einem Blogpost dargestellt (Bryan [2017](#)).

- inst

R Pakete werden in der Regel mit `install.packages()` installiert und werden irgendwo auf dem Computer auf dem R verwendet wird, abgespeichert. Sollen Dateien, welche nicht direkt in Zusammenhang mit dem Paket stehen mitgeliefert werden, dann ist der *inst* Ordner ein guter Ort dafür. Der Ordner wird im Stammordner des installierten Pakets gespeichert und bleibt so immer in Bezug zum Paket. Der Ordner bietet auch Platz für Sandboxes oder um mit neuen Funktionen zu experimentieren.

Neben der Ordnerstruktur ist für die stringente Entwicklung und anschliessende Nutzung des Pakets eine einheitliche Syntax wichtig. Für `{biviz}` wird der *snake_case* Ansatz verwendet. Beim *snake_case* werden alle Begriffe in Kleinbuchstaben oder Zahlen verwendet und mit einem Unterstrich (`_`) verbunden. Neben der Darstellung ist auch der Inhalt wichtig. Gute Namen können bereits eine Form der Dokumentation sein und entsprechend beschreiben was sie repräsentieren. Als Faustregel gilt: Variablen sind Nomen und beschreiben was sie sind (`abfallart`). Funktionen sind Verben und beschreiben was sie machen (`plot_amounts_grouped`) (Wickham [2022](#)). Es hilft sich in Erinnerung zu rufen, dass der Code auch von Menschen gelesen und verstanden werden muss. Für den Computer ist es nicht relevant ob `function1()` oder `compute_max()` steht. Für einen Menschen ist es jedoch ein grosser Unterschied und macht den Code verständlicher. Entsprechend soll der Code für Menschen einfach und verständlich gehalten werden.

4.2 Tidy evaluation

Im Tidyverse Ecosystem wird *tidy evaluation* verwendet, was eine Spezialform des non-standard evaluation ist.¹ Für das Paket `{biviz}` wurde das *data-masking* und *tidy-selection* angewendet. Mit dem data-masking können Variablen in bestimmten Funktionen so verwendet werden, als ob sie in der Programmierung bzw. Funktionsumgebung sind. Es reicht dann `variable_x` zu verwenden anstatt `df$variable_x`. Mit data-masking verschwimmt die Abgrenzung zwischen env-variables (Variablen für das Programmieren, welche in R mit `<-` erstellt werden und Objekte sind) und data-variables (Statistische Variablen, welche in Datentabellen bzw. env-variables gespeichert sind) (Wickham, François u.a. [2022](#)). Vereinfacht gesagt, wird durch das data-masking während des Aufrufs der Funktion die spezifisch aufgerufene env-variable (im Normalfall eine Datentabelle) zur Programmierung und die data-variable zur env-variable.

¹Mehr Details zu non-standard evaluation im Kapitel *Metaprogramming* des Buches *Advanced R* (Wickham [2019](#)).

```
df <- tibble(variable_x = c(1, 1, 3), variable_y = c("a", "b", "c"))
```

```
## env-variable
```

```
df
```

```
# A tibble: 3 x 2
  variable_x variable_y
    <dbl> <chr>
1         1 a
2         1 b
3         3 c
```

```
## data-variable
```

```
df$variable_x
```

```
[1] 1 1 3
```

```
## ohne tidy evaluation
```

```
df[df$variable_x == 1 & df$variable_y == "a", ]
```

```
# A tibble: 1 x 2
  variable_x variable_y
    <dbl> <chr>
1         1 a
```

```
## mit tidy evaluation
```

```
filter(df, variable_x == 1 & variable_y == "a")
```

```
# A tibble: 1 x 2
  variable_x variable_y
    <dbl> <chr>
1         1 a
```

Das tidy evaluation Konzept vereinfacht die interaktive Datenanalyse. Beim Programmieren mit Tools die data-masking benutzen gilt es jedoch gewisse Herausforderungen zu meistern. Beim Schreiben von eigenen Funktionen mit data-variables müssen sie im Code speziell mit `{{ variable_x }}` (“`{{`” wird auch curly-curly genannt) aufgerufen werden. Ohne die Einschliessung in `{{` sucht R in der Funktionsumgebung bzw. Programmierungsumgebung nach einer env-variable `variable_x` und findet sie nicht, da `variable_x` eine data-variable ist, welche eine Spalte im `df` (env-variable) ist. `{{` ist ein Weiterleitungsoperator, wodurch Funktionen mit data-masking, welche in der selber geschriebenen Funktion verwendet werden, ihr Verhalten in die neue Umgebung mitnehmen (Henry und Wickham 2022b; Henry und Wickham 2022a).

```
## interaktive Datenanalyse
```

```
df |>
```

```
  summarise(max = max(variable_x))
```

```

# A tibble: 1 x 1
  max
<dbl>
1     3

## Funktionen programmieren
compute_max <- function(data, variable) {
  data |>
    summarise(max = max({{ variable }}))
}

compute_max(df, variable_x)

# A tibble: 1 x 1
  max
<dbl>
1     3

```

Beim tidy-selection können in gewissen Funktionen auf die Variablen anhand ihres Namens, Typs oder ihrer Position zugegriffen werden. Tidy-selection folgt den gleichen Prinzipien wie data-masking. Folglich gelten die gleichen Regeln wie oben beim data-masking beschrieben (Wickham, François u. a. [2022](#)).

5 Workflow für die Entwicklung von Paketen in R

Wie für viele andere Aufgaben gibt es in R Pakete, welche die Entwicklung von eigenen Paketen unterstützen. Diese Pakete werden im `{devtools}` (Wickham, Hester u. a. [2022b](#)) zusammengefasst. Nachdem das Paket `{devtools}` installiert und geladen ist, kann der Entwicklungsprozess beginnen. Der hier beschriebene Ablauf orientiert sich am Kapitel *The Whole Game* der zweiten Edition des Buches *R Packages* (Wickham [2015a](#); Wickham und Bryan [2022](#)) und wird beispielhaft am Toypaket `{dogorcat}` durchgespielt. Die zweite Edition ist noch nicht veröffentlicht, die work-in-progress Version steht jedoch [online](#) zur Verfügung. In den folgenden Schritten werden die Funktionsaufrufe mit Angabe des spezifischen Pakets angegeben, damit die Herkunft der Funktionen ersichtlich ist.

5.1 Schritt 1: Initiierung des Pakets

Als erstes wird mit `usethis::create_package("dogorcat")` im derzeitigen Arbeitsverzeichnis das Paket erstellen. Alternativ kann auch ein spezifischer Ordner verwendet werden `usethis::create_package("C:/Users/Andriu/Documents/dogorcat")`. Mit `usethis::use_git()` und `usethis::use_github()` oder via Tools > Version Control kann ein Git repository sowie die Verbindung zum GitHub repository erstellt werden.

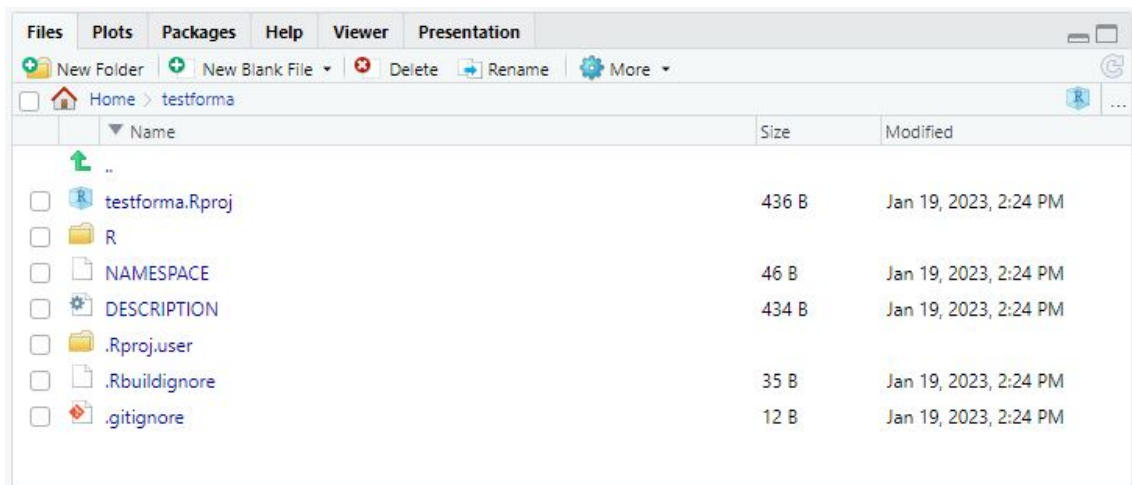


Abbildung 5.1: Initiierung des Pakets.

5.2 Schritt 2: Funktionen schreiben

Nach der Initiierung des Pakets können Funktionen geschrieben werden. Zuerst wird mit `usethis::use_r("dog_or_cat")` (kann später auch verwendet werden um das Skript zu öffnen) ein Skriptfile im Ordner R abgespeichert. Dieser Schritt kann natürlich auch manuell erfolgen. Anschliessend wird die Funktion geschrieben. Die Funktionen im {biviz} sind als Familien organisiert. Das heisst, pro Thema – beispielsweise Proportions – gibt es eine Datei, in der der Quellcode und die Dokumentation von mehreren Funktionen zusammen gefasst sind. Der Code kann direkt in der Funktion mit `#` dokumentiert werden. Wichtig beim Dokumentieren des Codes ist das *Warum*. Warum mache ich das so, wie ich es mache? Für weitere Details zum Schreiben von Funktionen eignet sich das Kapitel *Functions are for Humans and Computers* im Buch R for Data Science (Wickham und Grolemund 2016).

```
dog_or_cat <- function(favorite) {  
  # eine fehlerhafte eingabe soll so früh wie möglich erkannt werden  
  stopifnot("Das Argument 'favorite' muss ein String sein."  
    = is.character(favorite),  
    "Kein anderes Haustier möglich. Wähle Dog oder Cat."  
    = favorite == "dog" | favorite == "cat")  
  ifelse(favorite == "cat", "Miauu!", "Woof, woof!")  
}
```

Mit `devtools::load_all()` ("Ctrl + Shift + L" oder Build > Load All) kann anschliessend das Paket geladen werden und die Funktion steht zur Verfügung.

```
dog_or_cat("dog")
```

```
[1] "Woof, woof!"
```

Mit `devtools::check()` ("Ctrl + Shift + E" oder Build > Check Package) wird ein R CMD check aufgerufen, welcher prüft, ob das Paket funktionsfähig ist. Es ist zu empfehlen `devtools::check()` regelmässig auszuführen, damit frühzeitig Fehler entdeckt und behoben werden können. Beispielsweise nach dem Schreiben einer neuen Funktion.

5.3 Schritt 3: Dokumentation

Als erstes werden die Metadaten für das Paket erfasst. Dazu wird das File `DESCRIPTION` geöffnet und die relevanten Daten (Paketname, Autor, Licence, etc.) erfasst. Die Lizenz wird am einfachsten mit einem Shortcut wie `usethis::use_mit_licences()` ausgefüllt.

Anschliessend wird eine Dokumentation für die Funktion erstellt. Dazu wird das Paket {roxygen2} (Wickham, Danenberg u. a. 2022b) verwendet. Im Skript mit der Funktion wird der Cursor in der Funktion `dog_or_cat` platziert und via Code > Insert Roxygen Skeleton das Template aufgerufen. Anschliessend können alle Punkte bearbeitet werden.

```

#' Title
#'
#' @param favorite
#'
#' @return
#' @export
#'
#' @examples

```

```

dog_or_cat <- function(favorite) {
  # eine fehlerhafte eingabe soll so früh wie möglich erkannt werden
  stopifnot("Das Argument 'favorite' muss ein String sein."
    = is.character(favorite),
    "Kein anderes Haustier möglich. Wähle Dog oder Cat."
    = favorite == "dog" | favorite == "cat")
  ifelse(favorite == "cat", "Miauu!", "Woof, woof!")
}

```

Nun soll der Dokumentationsteil noch als separate Datei im Ordner *man* gespeichert werden, damit es später mit `?dog_or_cat` oder `help("dog_or_cat")` zur Verfügung steht. Mit `devtools::document()` (Ctrl + Shift + D oder Build > Document) wird in *man* die Datei `dog_or_cat.Rd` erstellt.

Die Verwendung von Funktionen aus anderen Paketen muss ebenfalls dokumentiert werden. Dazu wird mit `usethis::use_package("pkgname")` das Paket in `DESCRIPTION` als Import angegeben und mit `@importFrom` in der Dokumentation vermerkt. Alternativ kann auch die Funktion mit `pkgname::functionname` aufgerufen werden, wodurch das `@importFrom` hinfällig ist.

```

#' Lieblingshaustier eruieren
#'
#' @param favorite
#'
#' @return Ein character (string) Vektor.
#' @export
#'
#' @importFrom dplyr
#' if_else
#'
#' @examples
#' dog <- tidy_dog_or_cat("dog")
#' dog

```

```

tidy_dog_or_cat <- function(favorite) {
  # eine fehlerhafte eingabe soll so früh wie möglich erkannt werden
  stopifnot("Das Argument 'favorite' muss ein String sein."
    = is.character(favorite),
    "Kein anderes Haustier möglich. Wähle Dog oder Cat."
    = favorite == "dog" | favorite == "cat")
}

```



```

  if_else(favorite == "cat", "Miauu!", "Woof, woof!")
}

```

Als letztes wird noch ein README für GitHub benötigt. Hier werden der Zweck des Pakets, die Installationsanweisungen und einen kleinen Einblick in die Anwendung des Pakets gegeben. Natürlich können noch weitere Punkte ins README gepackt werden. Am einfachsten wird mit `devtools::use_readme_rmd()` ein RMarkdown Dokument erstellt, so kann die Seite in gewohnter Umgebung geschrieben werden. Anschliessend wird mit `devtools::build_readme()` das README.md für GitHub erstellt. Das README ist die Homepage und unterstützt beim Einstieg in das Paket. Später kann auch eine eigene Website erstellt werden.

5.4 Schritt 4: Testen

Das Testen der Funktionen ist ein wichtiger Teil der Paketentwicklung. Es ist so wichtig, dass es sich ab einer gewissen Relevanz (Anzahl Funktionen, Anzahl Nutzende, etc.) lohnt, die Tests zu formalisieren. Mit `{testthat}` (Wickham 2011) liefert das Paket `{devtools}` ein Framework, welches das Schreiben von Tests unterstützt. Für das Testen von grafischen Outputs wird zusätzlich noch das Paket `{vdiff}` (Henry, Pedersen u. a. 2023) benötigt. Beim Unit Test wird die erwartete Ausgabe eines Codes festgehalten und mit dem tatsächlichen Wert verglichen. Dieser Prozess läuft automatisch auf der Basis von Code. Beim Snapshot testing wird eine Datei erstellt, die für Menschen lesbar ist. Bei Datenvisualisierungen – wie im Paket `{biviz}` – wird das Snapshot testing angewendet. Hier wird ein Bildfile mit der erwarteten Darstellung erstellt. Beim Testen wird anschliessend der Output mit der erwarteten Darstellung abgeglichen. Snapshot tests für Grafiken sind anfällig. Aus diesem Grund, und da `{biviz}` noch in der Entwicklungsphase ist, werden die Tests bei `{biviz}` noch informell, also manuell gemacht. Ausgenommen davon sind die standardisierten R CMD checks wie in Kapitel 5.2 beschrieben. Beim nächsten Entwicklungsschritt sollen die Tests, wie oben beschrieben, formalisiert werden.

5.5 Schritt 5: Installieren

Vor dem Installieren des Pakets wird nochmals mit `devtools::check()` alles geprüft und anschliessend mit `devtools::install()` (oder von GitHub mit `devtools::install_github()`) installiert. Danach wird das Paket, wie alle Pakete, mit `base::library()` verwendet.

Die effektive Entwicklung von Paketen geschieht in den Schritten zwei bis vier. Die Anwendung und Reihenfolge dieser Schritte sind dabei dynamisch.

6 Verwendung von `{biviz}`

Mit `{biviz}` können häufig verwendete Datenvisualisierungen mit einfachen Funktionen erstellt werden. `{biviz}` stellt insbesondere Datenvisualisierungen, die oft im Bereich Business Intelligence (BI) verwendet werden, zur Verfügung. Das Paket implementiert Trends der Datenvisualisierung, wodurch sich das Paket auch für andere Zwecke eignet. In `{biviz}` werden `{ggplot2}` Wrapper zur Verfügung gestellt, welche die gängigen Datenvisualisierungen erzeugen. Dies hat den Vorteil, dass bei Standardgrafiken die Datenvisualisierung nicht Schicht für Schicht programmiert wird, wie es beim `{ggplot2}` Framework vorgesehen ist. Eine Zeile Code reicht, um eine Datenvisualisierung zu erstellen. Da das Paket auf `{ggplot2}` aufbaut, kann das erzeugte Objekt zu einem späteren Zeitpunkt dennoch angepasst werden. Um die Wrapper schlank zu halten, werden bei den meisten Funktionen in `{biviz}` die Daten vor der Übergabe in die Funktion so aufbereitet, dass keine Berechnungen innerhalb der Visualisierungsfunktion vorgenommen werden müssen. Ausnahmen mit komplexen Berechnungen sind in der Dokumentation vermerkt.

6.1 Datenvisualisierungsfamilien

Die Visualisierungen sind in 4 Gruppen (*amounts*, *distributions*, *proportions* und *time series*) aufgeteilt und folgen immer der gleichen Syntax. Jede Funktion startet mit `plot_*()`, anschliessend wird die Gruppe definiert `plot_amounts_*()` und am Ende wird die Form angegeben `plot_amounts_grouped()`.

In der Gruppe *amounts* sind jene Datenvisualisierungen zusammengefasst, die Mengen abbilden. Dabei werden die Zahlenwerte von Kategorien bzw. deren Anzahl in einer Variable dargestellt. Die Kategorien können zusätzlich einer Gruppe zugeordnet werden. Dazu werden Variablen verwendet, welche die Gruppenzugehörigkeit definieren.

Bei *distributions* werden Verteilungen innerhalb einer Variable dargestellt. Dazu können unterschiedliche Methoden verwendet werden, welche je nach Ausgangslage unterschiedliche Vorteile mit sich bringen. Bei den Datenvisualisierungen zu Verteilungen lohnt es sich verschiedene Grafiken auszuprobieren, um ein möglichst akkurates Bild der Daten zu erhalten. Die Funktion `plot_distributions_raincloud()` verfolgt beispielsweise das Ziel, einen gesamthaften Überblick (inkl. Rohdaten) zu präsentieren. Dies passiert indem verschiedene Layers (Boxplot-, Violin- und Punktediagramm) übereinander gelegt werden. Bei grossen Datenmengen gibt es jedoch so viele Datenpunkte, dass das Punktediagramm keinen Mehrwert mehr liefert. Deshalb bietet die Funktion `plot_distributions_boxplot()` eine Alternative. Die Diskussion rund um statistisch robusteren und transparenten Ansätzen zur Datenvisualisierung, ist ein Disziplin übergreifendes Thema (vgl. Allen u. a. [2021](#); Hehman und Xie [2021](#)).

Für *proportions* gibt es unterschiedliche Formen der Darstellung, die verschiedene Vorteile haben. Bei gestapelten Balken oder Donutplots ist es klar ersichtlich, dass es Teilmengen eines Ganzen sind. Nebeneinander angeordnete Balken zeigen den relativen Unterschied besser und eignen sich zudem für viele Teilmengen. Werden die Proportionen von verschiedenen Variablen untersucht, dann eignen sich wiederum gestapelte Balken. Durch Visualisierungssysteme wie `{ggplot2}` lassen sich Proportionen separat als Teil der Gesamtmenge darstellen. Mit `plot_proportions_sidebyside_density1()` werden zwei Probleme gelöst: Erstens, ist das Verhältnis zur Gesamtmenge ersichtlich (im Gegensatz zum Balkendiagramm). Zweitens, hat jede Teilmenge eine Grundlinie wodurch die Mengen einfach verglichen werden können (im Gegensatz zu gestapelten Balkendiagrammen). Mit `plot_proportions_sidebyside_density2()` kann der relative Anteil zu einem bestimmten Zeitpunkt einfach bestimmt werden (Nussbaumer Knaflitz 2017, S. 47; Wilke 2020, S. 92-94).

Daten mit einem Zeitpunkt haben eine inhärente Reihenfolge (geordnete Richtung) und Liniendiagramme eignen sich, die zeitliche Ordnung darzustellen, da abgesehen vom Anfang und Ende alle Datenpunkte einen Vorgänger und Nachfolger haben. Die Familie *time series* bietet neben unterschiedlichen Formen von Liniendiagrammen auch Visualisierungen von Trends an. Die Darstellung von Trends hilft, übergeordnete Entwicklungen zu erkennen. Um verschiedene Methoden der Glättung und Trendbereinigung bereitzustellen, greifen die Funktionen auf das Paket `{tsbox}` (Sax 2021) zurück.

6.2 Vorteile

Damit das Handling der Funktion in `{biviz}` einfach bleibt, haben die Funktionen Einschränkungen bzw. können sie nicht die ganze Palette an Möglichkeiten, die in `{ggplot2}` zur Verfügung steht, anwenden. Wie im Kapitel 2 gesehen, sind für ein effektives Data Storytelling eine flexible Anpassung von Datenvisualisierungen wichtig. Durch den modularen Aufbau von `{ggplot2}` geht dies relativ einfach. Mit dem Aufruf einer `{biviz}` Funktion wird ein `{ggplot2}` Objekt erstellt. Dieses Objekt kann mit `{ggplot2}` Funktionen angepasst und verfeinert werden. Geht eine Feinpolierung über die Möglichkeit des `{biviz}` hinaus, so kann dem erstellten Objekt einfach `{ggplot2}` Funktionen angefügt werden, um Anpassungen an der Grafik zu machen. Wichtig: Normalerweise werden in R aufeinanderfolgende Funktionsaufrufe mit der Pipe `|>` verbunden. Bei `{ggplot2}` Objekten wird dafür das `+` verwendet.

```
library(ggplot2)

## raw plot
plot_abfall_zh_raw <-
  df_abfall_zh |>
  # biviz funktion
  plot_amounts_vertical(
    # die daten haben keine logische reihenfolge, deshalb werden sie
    # der größe nach sortiert. dies geschieht mit fct_reorder
    x = forcats::fct_reorder(Gemeinde, Wert),
    y = Wert
```

```

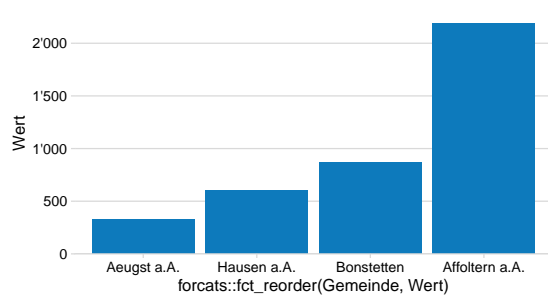
)

## make the plot nice
plot_abfall_zh_nice <-
  # das vorherige ggplot2 objekt wird verwendet
  plot_abfall_zh_raw +
  # kontext hinzufügen
  ggtitle("Anzahl Brennbare Abfälle und Sperrgut\nje Gemeinde im Jahr 2021") +
  labs(
    x = "Gemeinde",
    y = "Menge in Tonnen"
  ) +
  theme(legend.position = "none")

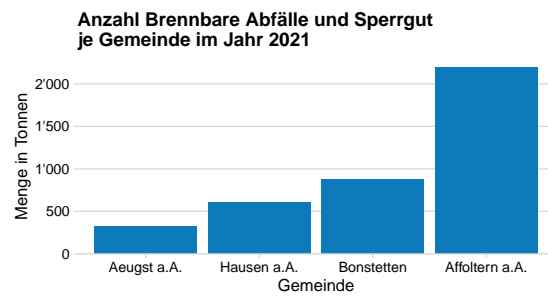
## let the plot shine
plot_abfall_zh_shine <-
  # das vorherige ggplot2 objekt wird verwendet
  plot_abfall_zh_nice +
  # mit diesem schritt werden alle balken ausser, der hervorzuhebende grau
  # "übermalt" indem ein neuer layer auf die visualisierung gelegt wird
  geom_col(
    data = filter(df_abfall_zh,
                  abfall_pro_person != min(df_abfall_zh$abfall_pro_person)),
    mapping = aes(
      x = fct_reorder(Gemeinde, Wert),
      y = Wert
    ),
    fill = "lightgrey",
    position = "dodge"
  ) +
  ggtitle(
    # mit dem paket ggtext kann html/markdown code im text verwendet werden
    paste0(
      "<span style = 'color:lightgrey;'>Im Jahr 2021 hatte die</span><br>",
      "<span style = 'color:#0d7abc; style = font-size:24pt'"
      Gemeinde Affoltern a.A.</span><br>",
      " total am meisten ",
      "<span style = 'color:lightgrey;'>brennbare Abfälle und Sperrgut</span><br>",
      "<span style = 'color:lightgrey;'>aber</span>",
      " pro Kopf am wenigsten (5.6 Tonnen)"
      # "<span style = 'color:lightgrey;'>Menge</span>"
    )
  ) +
  labs(y = "Totale Abfallmenge\nin Tonnen") +
  # damit der html/markdown code im text gerendert werden kann, benötigt es
  # die funktion element_markdown
  theme(
    plot.title = ggtext::element_markdown(size = 14),

```

```
axis.title.x = element_blank(),
axis.title.y = element_text(size = 11, vjust = 3)
)
```

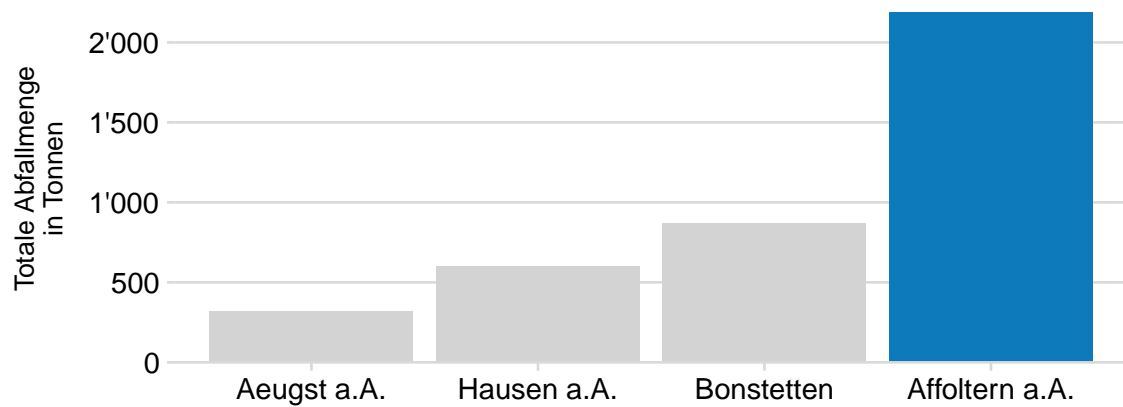


(a) Raw Plot



(b) Make the Plot nice

Im Jahr 2021 hatte die
Gemeinde Affoltern a.A.
 total am meisten brennbare Abfälle und Sperrgut
 aber pro Kopf am wenigsten (5.6 Tonnen)



(c) Let the Plot shine

Abbildung 6.1: Verfeinern von biviz Visualisierungen.

7 Fazit

Der Ausgangspunkt der Masterarbeit war die fehlende Qualität von Datenanalysen bei nicht reproduzierbaren Workflows und fehlendes Storytelling. Der Fokus lag auf der praktischen Umsetzung der Kommunikation mit Daten. Damit das Data Storytelling für die Anwendenden möglichst einfach gestaltet werden kann, wurde das R-Paket `{biviz}` entwickelt. Durch dieses Paket wurde im gleichen Zuge auch die zweite Problemstellung angegangen. Das Paket wird in der Skript basierten Programmierumgebung von R angewendet, wodurch jeder Arbeitsschritt nachvollziehbar ist. Zudem könnte auch eine Datenpipeline von den Rohdaten bis zum finalen Report gebaut werden.

Im ersten Teil der Arbeit wurde aufgezeigt, wie aus Datenwerten eine Visualisierung entsteht. Für diesen Prozess ist die Beziehung zwischen Daten, Skalen und Aesthetics zentral. Sprich die Verbindung zwischen Daten und quantifizierbaren Merkmalen, in Form von abbildbaren Elementen (Position, Grösse, Farbe und Form). Zudem wurden wichtige Aspekte des Data Storytellings, wie die Entwicklung und Steuerung einer Geschichte, aufgezeigt. Anschliessend wurde die technische Basis für die Entwicklung von `{biviz}` dargelegt und erläutert. Im Mittelpunkt lag dabei die Konzepte von Tidy Data und Grammar of Graphics. Tidy Data heisst, alle Variablen sind Spalten, alle Untersuchungen sind Zeilen und alle Werte sind Zellen. Durch diese Ordnung der Daten sind die weiteren Analyseschritte einfacher umzusetzen und sie gibt eine konsequente Struktur vor. Eine konsequente Datenstruktur hilft Tools zu entwickeln, da ein Standard des Dateninputs vorliegt. Mit ihren Stages und die damit verbundene Zerstückelung der Erstellung von Grafiken, bietet die Grammar of Graphics eine grosse Flexibilität bei der Datenvisualisierung. Die strukturierte Beziehung von Datenvariablen und deren Repräsentation wurde mit dem Paket `{ggplot2}` in `{biviz}` integriert. Im dritten und vierten Teil wurde die effektive Entwicklung von `{biviz}` beispielhaft dargestellt und auf Herausforderungen und Probleme eingegangen. Zum einen wurde mit der Ordnerstruktur und Codesyntax ein Gerüst für den Entwicklungsprozess definiert. Zum anderen wurden die Inhalte und Funktionen (Dokumentation, Quellcode, etc.) kurz dargelegt. Da `{biviz}` auf dem Paket `{ggplot2}` basiert, welches tidy evaluation unterstützt, wurden die Unterscheidung zwischen env-variables und data-variables eingeführt. Mit den curly-curly (“`{{}}`”) bietet das Tidyverse Framework einen intuitiven Umgang mit diesem Problem beim Erstellen eines Pakets. Der Entwicklungsworkflow besteht aus vier Schritten, welcher jeweils kleinere Teilschritte beinhaltet. Die effektive Entwicklung findet dabei in den Schritten “Funktionen schreiben”, “Dokumentation” und “Testen” statt. Eine genaue Abfolge zwischen diesen Schritten gibt es nicht. Wichtig ist es, dass die Tests und Dokumentationen regelmässig durchgeführt bzw. ergänzt werden. Es muss noch nicht die Schlussversion sein. Solange der geschriebene Code und die Gedanken dazu noch frisch sind, ist es einfacher, entsprechende Tests und Dokumentationen zu schreiben. Das letzte Kapitel zeigte die Anwendung des Pakets und ihre Vorteile. Dafür wurden

die Funktionsfamilien beschrieben und ein Showcase zur Anwendung von `{biviz}` präsentiert.

Datenvisualisierung ist teils Kunst und teils Wissenschaft. Die Herausforderung besteht darin, die Kunst richtig zu machen, ohne die Wissenschaft falsch zu machen, und umgekehrt. (Wilke 2020, S. 1)

Die Arbeit hatte das Ziel, ein Tool zu entwickeln, welches das Data Storytelling und die Datenvisualisierungen vereinfacht, aber dennoch eine grosse Flexibilität bereithält. Mit `{biviz}` werden schnell Basisgrafiken erstellt und es steht mehr Zeit zur Verfügung, um sich dem Data Storytelling zu widmen. Gleichzeitig wird mit der Verwendung von `{biviz}` ein minimaler Standard für die Grafiken eingeführt. Dadurch trägt `{biviz}` dazu bei, sowohl die Kunst als auch die Wissenschaft richtig zu machen, ein bisschen einfacher geworden ist. Die Fähigkeit eines “Auge” kann das Paket aber nicht kompensieren. Folglich bleibt die Auseinandersetzung und Optimierung von Datenvisualisierungen ein lebenslanger Prozess. In diesem Sinne ist `{biviz}` kein statisches Projekt, sondern wird in Zukunft weiterentwickelt und optimiert.

`{biviz}` importiert einige Funktionen aus anderen R-Paketen, wodurch sich Abhängigkeiten ergeben. Die grosse Mehrheit der Pakete kommen aus dem Tidyverse Framework, welches vom Unternehmen Posit unterstützt wird und entsprechend eine gewisse Langlebigkeit und Support garantiert. Andere Pakete, wie das `{colorspace}` (Zeileis u. a. 2020), sind in einem universitären Kontext entstanden, was ebenfalls eine Beständigkeit gewährleistet. Der nächste wichtige Schritt für `{biviz}`, ist die Implementierung von formellen Tests. Zudem können weitere Datenvisualisierungen (Heatmap, Punktediagramm, Lollipopdiagramm, etc.) ergänzt und der Code optimiert werden. Ein weiteres Ziel bei der Weiterentwicklung des Pakets ist es, dass die Basisgrafiken optimierter für das Data Storytelling sind.

Das Paket steht noch in seinen Kinderschuhen und hat somit Steigerungspotential. Der Arbeit liegt das Motto von Hadley Wickham zugrunde:

The only way to write good code is to write tons of shitty code first. Feeling shame about bad code stops you from getting to good code. (Wickham 2015b)

In diesem Sinne ist `{biviz}`, wie es aktuell dasteht, ein erster Entwurf, welcher in Zukunft stetig eine Weiterentwicklung erfährt.

Literaturverzeichnis

- Allen, Micah u. a. (2021). *Raincloud plots: a multi-platform tool for robust data visualization*. Techn. Ber. DOI: [10.12688/wellcomeopenres.15191.2](https://doi.org/10.12688/wellcomeopenres.15191.2).
- Bannert, Matthias (2022). *DevOps Carpentry*. URL: <https://devops-carpentry.github.io/book/> (besucht am 30.01.2023).
- Bartram, Lyn, Abhisekh Patra und Maureen Stone (2017). “Affective Color in Visualization”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Denver Colorado USA: ACM, S. 1364–1374. DOI: [10.1145/3025453.3026041](https://doi.org/10.1145/3025453.3026041).
- Borland, David und Russell M. Taylor Ii (2007). “Rainbow Color Map (Still) Considered Harmful”. In: *IEEE Computer Graphics and Applications* 27.2, S. 14–17. DOI: [10.1109/MCG.2007.323435](https://doi.org/10.1109/MCG.2007.323435).
- Bryan, Jennifer (2017). *Project-oriented workflow*. URL: <https://www.tidyverse.org/blog/2017/12/workflow-vs-script/> (besucht am 16.01.2023).
- Cleveland, William S. und Robert McGill (1984). “Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods”. In: *Journal of the American Statistical Association* 79.387, S. 531–554. DOI: [10.1080/01621459.1984.10478080](https://doi.org/10.1080/01621459.1984.10478080).
- Diaz-Bone, Rainer (2006). *Statistik für Soziologen*. UTB für Wissenschaft : Uni-Taschenbücher. Konstanz: UVK Verlagsgesellschaft.
- Dykes, Brent (2020). *Effective data storytelling: how to drive change with data, narrative and visuals*. Hoboken, New Jersey: John Wiley und Sons, Inc.
- Field, Andy, Jeremy Miles und Zoë Field (2012). *Discovering Statistics using R*. Los Angeles, London, New Delhi, Singapore, Washington DC: Sage.
- Gerste, Ronald D. (2022). *Die Heilung der Welt: das Goldene Zeitalter der Medizin 1840-1914*. Stuttgart: Klett-Cotta.
- Hawkins, Ed (2015). “Scrap rainbow colour scales”. In: *Nature* 519.7543, S. 291–291. DOI: [10.1038/519291d](https://doi.org/10.1038/519291d).
- Healy, Kieran (2018). *Data visualization: a practical introduction*. Princeton, NJ: Princeton University Press.
- Helman, Eric und Sally Y. Xie (2021). “Doing Better Data Visualization”. In: *Advances in Methods and Practices in Psychological Science* 4.4. Publisher: SAGE Publications Inc, S. 1–18. DOI: [10.1177/25152459211045334](https://doi.org/10.1177/25152459211045334).
- Henry, Lionel, Thomas Lin Pedersen u. a. (2023). *vdiffr: Visual Regression Testing and Graphical Diffing*. <https://github.com/r-lib/vdiffr>. URL: <https://r-lib.org/reference/topic-data-mask-programming.html> (besucht am 23.01.2023).
- Henry, Lionel und Hadley Wickham (2022a). *Data mask programming patterns*. <https://github.com/r-lib/rlang>. URL: <https://r-lib.org/reference/topic-data-mask-programming.html> (besucht am 16.01.2023).
- (2022b). *What is data-masking and why do I need it?* <https://github.com/r-lib/rlang>. URL: <https://r-lib.org/reference/topic-data-mask.html> (besucht am 16.01.2023).

- Kanizsa, Gaetano (1970). “Amodale Ergänzung und ”Erwartungsfehler” des Gestaltpsychologen”. In: *Psychologische Forschung* 33.4, S. 325–344. DOI: [10.1007/BF00424558](https://doi.org/10.1007/BF00424558).
- Lowndes, Julie und Allison Horst (2020). *Tidy data for efficiency, reproducibility, and collaboration*. URL: <https://www.openscapes.org/blog/2020/10/12/tidy-data/> (besucht am 30. 12. 2022).
- Nussbaumer Knaflic, Cole (2017). *Storytelling mit Daten: die Grundlagen der effektiven Kommunikation und Visualisierung mit Daten*. München: Verlag Franz Vahlen.
- Okabe, Masataka und Kei Ito (2008). *Color Universal Design (CUD) / Colorblind Barrier Free*. URL: <https://jfly.uni-koeln.de/color/#pallet> (besucht am 27. 01. 2023).
- Sax, Christoph (2021). *tsbox: Class-Agnostic Time Series in R*. <https://github.com/christophsax/tsbox>. URL: <https://www.tsbox.help> (besucht am 23. 01. 2023).
- Silver, Nate (2020). *The signal and the noise: why so many predictions fail - but some don't*. Published with a new preface in Penguin Books 2020. New York, NY: Penguin Books.
- Snow, John (1855). *On the Mode of Communication of Cholera. 2. Auflage*. London.
- Spiegelhalter, David (2020). *The art of statistics: learning from data*. Paperback edition. UK: Pelican Books.
- Tufte, Edward R. (2001). *The visual display of quantitative information*. 2nd ed. Cheshire, Conn: Graphics Press.
- Tukey, John Wilder (1977). *Exploratory data analysis*. Reading, Mass: Addison-Wesley Pub. Co.
- Wickham, Hadley (2010). “A Layered Grammar of Graphics”. In: *Journal of Computational and Graphical Statistics* 19.1, S. 3–28. DOI: [10.1198/jcgs.2009.07098](https://doi.org/10.1198/jcgs.2009.07098).
- (2011). “testthat: Get Started with Testing”. In: *The R Journal* 3.1. <https://testthat.r-lib.org/>, S. 5. DOI: [10.32614/RJ-2011-002](https://doi.org/10.32614/RJ-2011-002).
- (2014). “Tidy Data”. In: *Journal of Statistical Software* 59, S. 1–23. DOI: [10.18637/jss.v059.i10](https://doi.org/10.18637/jss.v059.i10).
- (2015a). *R packages*. First edition. OCLC: ocn898161451. Sebastopol, CA: O’Reilly Media.
- (17. Apr. 2015b). *The only way to write good code is to write tons of shitty code first...* URL: <https://twitter.com/hadleywickham/status/589068687669243905> (besucht am 20. 01. 2023).
- (2016). *Ggplot2*. New York, NY: Springer Science+Business Media, LLC.
- (2019). *Advanced R*. Second edition. Boca Raton: CRC Press/Taylor und Francis Group.
- (2022). *The tidyverse style guide*. <https://github.com/tidyverse/style>. URL: <https://style.tidyverse.org> (besucht am 27. 01. 2023).
- Wickham, Hadley und Jennifer Bryan (2022). *R Packages (2e)*. URL: <https://r-pkgs.org/> (besucht am 10. 02. 2023).
- Wickham, Hadley, Peter Danenberg u. a. (2022a). *roxygen2: In-Line Documentation for R*. <https://github.com/r-lib/roxygen2>. URL: <https://roxygen2.r-lib.org/>.
- (2022b). *roxygen2: In-Line Documentation for R*. <https://github.com/r-lib/roxygen2>. URL: <https://roxygen2.r-lib.org/> (besucht am 23. 01. 2023).
- Wickham, Hadley, Romain François u. a. (2022). *Programming with dplyr*. <https://github.com/tidyverse/dplyr>. URL: <https://dplyr.tidyverse.org/articles/programming.html> (besucht am 16. 01. 2023).
- Wickham, Hadley und Garrett Golemund (2016). *R for data science: import, tidy, transform, visualize, and model data*. First edition. OCLC: ocn968213225. Sebastopol, CA: O’Reilly.

- Wickham, Hadley, Jim Hester u. a. (2022a). *devtools: Tools to Make Developing R Packages Easier*. <https://github.com/r-lib/roxygen2>. URL: <https://devtools.r-lib.org/> (besucht am 27.01.2023).
- (2022b). *devtools: Tools to Make Developing R Packages Easier*. <https://github.com/r-lib/devtools>. URL: <https://devtools.r-lib.org/> (besucht am 23.01.2023).
- Wilke, Claus (2020). *Datenvisualisierung - Grundlagen und Praxis: wie Sie aussagekräftige Diagramme und Grafiken gestalten*. 1. Auflage. Heidelberg: O'Reilly.
- Wilkinson, Leland (2005). *The Grammar of Graphics*. 2nd ed. Statistics and computing. New York: Springer.
- (2012). “The Grammar of Graphics”. In: *Handbook of Computational Statistics*. Hrsg. von James E. Gentle, Wolfgang Karl Härdle und Yuichi Mori. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 375–414. DOI: [10.1007/978-3-642-21551-3_13](https://doi.org/10.1007/978-3-642-21551-3_13).
- Zeileis, Achim u. a. (2020). “colorspace : A Toolbox for Manipulating and Assessing Colors and Palettes”. In: *Journal of Statistical Software* 96.1. DOI: [10.18637/jss.v096.i01](https://doi.org/10.18637/jss.v096.i01).