Erschienen in: Bubenhofer, Noah/Kupietz, Marc (Hrsg.):
Visualisierung sprachlicher Daten. Visual Linguistics - Praxis - Tools. -
Heidelberg: Heidelberg University Publishing, 2018. S. 227-256.

Sascha Wolfer / Sandra Hansen-Morath

Visualisierung sprachlicher Daten mit R

Abstract Die ansprechende und geeignete Visualisierung linguistischer Daten gewinnt analog zum steigenden Einfluss quantitativer Methoden in der Linguistik immer mehr an Bedeutung. R ist eine flexible und freie Entwicklungsumgebung zur Umsetzung von statistischen Analysen, die zahlreiche Optionen zur Datenvisualisierung bereithält und sehr gut für große Datensätze geeignet ist. Statistische Analysen und Visualisierungen von Daten werden auf diese Weise in einer Umgebung verzahnt. Durch die zahlreichen Zusatzpakete stehen auch weiterhin zeitgemäße Methoden zur Verfügung, um (linguistische) Daten zu analysieren und darzustellen.
Unser Beitrag vermittelt einen stark anwendungsorientierten Einstieg in das Programm und legt mithilfe von vielen praktischen Übungen und Anwendungsbeispielen die Grundlagen für ein eigenständiges Weiterentwickeln der individuellen Fähigkeiten im Umgang mit der Software.
Neben einer kurzen, eher theoretisch angelegten Einleitung zu explorativen und explanatorischen Visualisierungsstrategien von Daten werden verschiedene Pakete vorgestellt, die für die Visualisierung in R benutzt werden können.

1. Einleitung

Mit dem wachsenden Einfluss quantitativer Ansätze in der Linguistik gewinnt die ansprechende und geeignete Visualisierung linguistischer Forschungsergebnisse kontinuierlich an Bedeutung. Die im folgenden Beitrag vorgestellte freie Statistikumgebung R (R Core Team 2016) bietet eine solche Möglichkeit, statistische Analyse und deren Visualisierung zu verzahnen. Dank der fortlaufenden Entwicklung zahlreicher, den Funktionsumfang von R erweiternder Zusatzpakete ist davon auszugehen, dass Forscherinnen und Forschern auch in Zukunft zeitgemäße Methoden zur Verfügung stehen werden, linguistische Daten zu analysieren und darzustellen.

Im Vergleich zu den anderen Beiträgen in diesem Band fällt dieser Beitrag insofern aus dem Rahmen, als er in einem tutorialartigen Duktus gehalten ist. Dies ist dem Umstand geschuldet, dass der Beitrag in der Tat aus einem Tutorial

Erschienen in: Noah Bubenhofer u. Marc Kupietz (Hrsg.): Visualisierung sprachlicher Daten. Heidelberg: Heidelberg University Publishing, 2018
DOI: https://doi.org/10.17885/heiup.345.474

entstanden ist, das wir auf dem Herrenhäuser Symposium „Visuelle Linguistik“ in Hannover gehalten haben – jener Veranstaltung, aus der dieser Band hervorging. Da es das Ziel des Tutorials wie auch dieses Beitrags ist, einige Visualisierungsoptionen innerhalb von R vorzustellen und anhand linguistischer Beispiele zu illustrieren, gehen wir nicht auf kommerzielle Software (bspw. SPSS, SAS oder Stata) oder quelloffene Alternativen zu R (bspw. bestimmte Anwendungsszenarien der Sprache Python) ein. Zum Verständnis des Inhalts werden bei der Leserin / dem Leser zumindest erste Programmiererfahrungen (idealerweise in R selbst) vorausgesetzt.

1.1 Explorative und explanatorische Visualisierung von Daten

Visualisierungen sind meist entweder explorativ oder explanatorisch. Explorative Visualisierungen sind all jene Schaubilder, Graphen o. Ä., die Forschende einsetzen, um erhobene Daten selbst besser kennenzulernen und/oder Zusammenhänge zu begreifen bzw. zu entdecken. Dabei liegt der Fokus zunächst nicht auf einer ästhetisch ansprechenden Form. Auch auf eine extensive Annotation der Grafiken wird bei explorativen Visualisierungen im Allgemeinen verzichtet. Da die Forschenden die Schaubilder, Graphen etc. selbst entworfen haben, kennen sie ja die Bedeutung bspw. darin vorkommender Achsen oder Datenpunkte. Explorative Visualisierungen sind nicht unbedingt mit Verfahren der explorativen Statistik (z. B. Cluster- oder Hauptkomponentenanalysen) gleichzusetzen. Auch ein simples Streudiagramm (vgl. bspw. Abb. 1) kann explorativen Zwecken dienen. Wie bereits erwähnt: Primäres Ziel explorativer Visualisierungen ist es, die eigenen Daten besser kennenzulernen. In Zielgruppen gesprochen sind explorative Grafiken somit vom forschenden Individuum für sich selbst oder für Kolleginnen und Kollegen, die mit den Daten sehr gut vertraut sind.

Explanatorische Visualisierungen sind dagegen auf eine Zielgruppe ausgerichtet, die mit den zugrunde liegenden Daten nicht oder nur sehr wenig vertraut ist. Daher müssen meist unmissverständliche Achsenbeschriftungen verwendet und gegebenenfalls zusätzliche Annotationen (Legenden, Beschriftungen von Datenpunkten) hinzugefügt werden. Das Ziel explanatorischer Visualisierungen ist es, den Rezipient/innen Sachverhalte bzw. Wissen zu vermitteln oder sie von etwas zu überzeugen. Rezipient/innen explanatorischer Grafiken können Fachkolleg/innen (bei wissenschaftlichen Publikationen) sein, aber auch Laien, beispielsweise bei einer Infografik im Web oder einem Schaubild in einer populärwissenschaftlichen Zeitschrift oder Zeitung.

1.2 Von Daten zu Schaubildern

Bei der Frage, welche Art von Visualisierung sich am besten eignet, sollte immer von den Daten ausgegangen werden, die zu visualisieren sind. Einige einfache Fragen können helfen, die Gruppe von Visualisierungen zumindest einzuschränken. Eine erste Frage könnte lauten: Liegen mir Daten vor, in denen Informationen über *Einzelfälle* vorliegen, oder handelt es sich um auf welche Weise auch immer *aggregierte* Daten? So stellt beispielsweise eine Kontingenztabelle, die Korpushäufigkeiten in Bezug auf verschiedene Kombinationen von Merkmalen enthält, eine aggregierte Datenstruktur dar, weil aus der Tabelle nicht mehr auf den konkreten Einzelfall zurückgeschlossen werden kann (siehe unter anderem Tabelle 2). Eine Tabelle hingegen, in der jede individuelle sprachliche Einheit, sei es ein Wort, eine Phrase, eine Konstruktion oder ein Satz, in einer eigenen Zeile repräsentiert ist und deren Spalten Informationen zu dieser konkreten Einheit (Korpushäufigkeit oder -quelle, Wortart, syntaktische Einbettungstiefe etc.) beinhalten, weist keine aggregierte Datenstruktur auf. Grundsätzlich können auf der Grundlage von nicht-aggregierten, also einzelfallbasierten Datenstrukturen mehr Visualisierungen erstellt werden, da mehr Informationen vorhanden sind, nämlich zu jedem Einzelfall. Ohne diese Einzelfallinformationen sind zum Beispiel keine logistischen Regressionsanalysen möglich, mit denen man das Eintreten oder Ausbleiben von Ereignissen auf der Basis von einem oder mehreren Prädiktor(en) vorhersagen kann. Aggregierte Datenstrukturen können außerdem bei Bedarf aus Einzelfällen jederzeit abgeleitet werden.

Außer der Art der Datenstruktur ist natürlich auch das Ziel der Visualisierung ausschlaggebend für die Wahl des Diagramms. Wenn ich den Unterschied zwischen zwei Gruppen zeigen möchte, könnte ein Balkendiagramm, evtl. mit Fehlerindikatoren, angebracht sein. Möchte ich die Aufmerksamkeit auf den zeitlichen Verlauf lenken, wäre ein Liniendiagramm geeigneter. Wenn ich auf der Suche nach statistischen Ausreißern bin oder einfach einen Eindruck von der Verteilung gewinnen möchte, bietet sich ein Boxplot oder ein Histogramm an.

Als dritter Faktor ist auch die Anzahl der an der Visualisierung beteiligten Variablen relevant: Wie viele Dimensionen müssen im Schaubild abgetragen werden, um das zu visualisieren, was ich zeigen möchte? Ist genau eine Variable beteiligt, handelt es sich meist um Visualisierungen von Verteilungen. Dazu gehören u a. Histogramme, Dichtekurven, Boxplots, Violin-Plots *(violin plots).*[1] Soll die Beziehung zwischen zwei Variablen gezeigt werden, kommt es darauf an, wie die Variablen jeweils skaliert sind. So eignet sich z. B. ein Balkendiagramm

1 Violin-Plots sind Boxplots recht ähnlich. Dabei werden anstatt (oder zusätzlich zu) einer Box links und rechts Dichtekurven angezeigt. In R sind Violin-Plots in den Paketen „ggplot2“ (siehe Abschnitt 3.1) und „vioplot“ verfügbar.

für die Darstellung des Zusammenhangs zwischen einer kategorialen (nominal- oder ordinalskalierten) und einer kontinuierlichen (intervallskalierten) Variable. Gruppiert oder stapelt man Balkendiagramme, kann noch eine weitere kategoriale Variable berücksichtigt werden. Zwei kontinuierliche Variablen und eine kategoriale Variable können in einem Streudiagramm im Zusammenhang betrachtet werden. Dazu werden die beiden kontinuierlichen Variablen auf der x- und y-Achse abgetragen, während die kategoriale Variable beispielsweise durch Farbe oder Form der Datenpunkte dargestellt wird. Tabelle 1 gibt einen Überblick über einige mögliche Kombinationen von Variablen und möglichen Visualisierungen.

Tabelle 1: Anzahl zu visualisierender kategorialer und kontinuierlicher Variablen und einige einfache passende Diagrammarten.

Kategoriale Variablen	**Kontinuierliche Variablen**	**Mögliche Visualisierung(en)**
1	0	Balkendiagramm, Tortendiagramm
> 1	0	Mosaikplot, Assoziationsplot, Gruppiertes/gestapeltes Balkendiagramm
0	1	Dichtekurve, Histogramm, Boxplot, Violin-Plot
0	2	Streudiagramm
0	> 2	3D-Diagramm, Streudiagramm + Radius der Datenpunkte
1	1	Balkendiagramm, Liniendiagramm, Gruppierte Boxplots
1	2	Streudiagramm + Farbe/Form der Datenpunkte
2	1	Gruppiertes Balkendiagramm, Heatmap, Hexbin-Plot

Tabelle 1 enthält längst nicht alle Arten von Visualisierungen, die mit den entsprechenden Kombinationen von Variablen möglich sind. Genannt sind aber die meisten grundlegenden Verfahren, die innerhalb der linguistischen Forschung etabliert sind.

1.3 Gliederung

Bevor wir uns nun konkret mit den Visualisierungsmöglichkeiten in R beschäftigen, sei an dieser Stelle noch ein kurzer Überblick über die Gliederung des folgenden Beitrags gegeben. In Kapitel 2 werden wir uns mit einigen R-Paketen beschäftigen, die für die Visualisierung in R benutzt werden können. In Abschnitt 2.1 gehen wir zunächst auf das in R bereits integrierte Paket „graphics“ ein. Dabei werden wir einerseits Plots selbst erstellen, andererseits aber auch zeigen, wie Elemente in bereits existierende Plots eingefügt werden können. Über diese Funktion können mächtige maßgeschneiderte Visualisierungen erstellt werden. In Abschnitt 2.2 widmen wir uns der Darstellung von kategorialen Daten und führen Mosaik- und Assoziationsplots ein, wie sie im Paket „vcd“[2] (Meyer 2015) verfügbar sind. In Abschnitt 2.3 zeigen wir, wie mit dem Paket „effects“ (Fox 2003) die Schätzer aus bereits vorhandenen statistischen Modellen extrahiert und abgetragen werden können. In Kapitel 3 besprechen wir zwei alternative Plot-Pakete: Mit dem Paket „ggplot2“ (Wickham 2009) können Visualisierungen basierend auf der „Grammar of Graphics“ (Wilkinson 20015) erstellt werden. Dieses Paket stellen wir (kurz) in Abschnitt 3.1 vor. Abschnitt 3.2 schließlich verwenden wir darauf, das R-Paket „rCharts“ (Vaidyanathan 2013) vorzustellen, mit dem interaktive Grafiken erstellt werden können, die auf der JavaScript-Bibliothek d3.js basieren. Ein Schlusskapitel rundet den Beitrag ab.

2. Visualisierung deskriptiver Statistiken und inferenzstatistischer Maße

2.1 Die Basisausstattung: Das Paket „graphics“

Bereits die Grafikfunktionen, die in R ohne jegliche Zusatzpakete zur Verfügung stehen, sind vielfältig anpassbar. Mit diesen „Bordmitteln“ können sowohl explorativ als auch explanatorisch ausgerichtete Visualisierungen erstellt werden. Benutzer/innen können dabei vorgefertigte Funktionen nutzen, um komplexe Grafiken zu erstellen. Da die Logik einer Art Baukastenprinzip folgt, können aber auch beliebige Elemente miteinander kombiniert und so einfache Grafiken sukzessive erweitert und an die eigenen Visualisierungsziele angepasst werden.

Einfache, bereits in Abschnitt 1.2 genannte Schaubilder können mit den Funktionen plot() für x-y-Streudiagramme und Liniendiagramme, barplot() für Balkendiagramme, hist() für Histogramme, pie() für Tortendiagramme und

2 Zusatzpakete können in R über die Funktion install.packages() installiert und über library() eingebunden werden.

boxplot() für Boxplots erstellt werden. Die Arbeitsweise und somit die grafischen Produkte dieser Funktionen können über Parameter angepasst werden. Eine Übersicht über alle Grafikparameter wie Ränder, Farben usw. kann in R über die Eingabe von ?par angefordert werden. Möchte man sich über die Parameter informieren, die jeder dieser Funktionen eigen sind, bspw. die Reichweite der Hinges[3] in Boxplots oder die Anzahl der Bins in Histogrammen, sollte man die Hilfeseite der jeweiligen Funktion (bspw. ?boxplot oder ?hist) zurate ziehen.

Das Interessante am Baukastenprinzip des Basispakets für Grafiken in R ist, dass man zu jedem bestehenden Plot Elemente hinzufügen kann. Alle oben genannten Funktionen starten in ihrer Standardeinstellung ein neues „Grafikgerät“ *(graphics device)*. Funktionen wie abline(), lines(), points(), curve(), arrows(), text() oder legend() fügen den Grafiken die entsprechenden Elemente in bereits geöffneten Grafikgeräten hinzu. Wir demonstrieren dies anhand eines Datensatzes zu einer lexikalischen Entscheidungsaufgabe, der im Paket „languageR“[4] (Baayen 2011) enthalten ist. Der Datensatz besteht aus 1659 Antworten auf eine lexikalische Entscheidungsaufgabe. Wir visualisieren den Zusammenhang zwischen Worthäufigkeit, semantischer Klasse und Reaktionszeit in der lexikalischen Entscheidungsaufgabe, d. h., wie lange jede Teilnehmerin/jeder Teilnehmer des Versuchs benötigt hat, um ein englisches Wort in Abhängigkeit von seiner semantischen Klasse (Tier/Pflanze) und seiner Vorkommenshäufigkeit als solches zu verifizieren. Die Nichtwörter sind nicht im Datensatz enthalten.

```
> library(languageR)
> lexdec$plot.col <- ifelse(lexdec$Class == "animal",
                            "#FF000099", "#00FF0099")
> plot(lexdec$Frequency, lexdec$RT, col = lexdec$plot.col,
       pch = 19, yaxt = "n", xaxt = "n",
       xlab = "Häufigkeit“, ylab = "Reaktionszeit (ms)",
       main = "Häufigkeit und Reaktionszeit in 'lexdec'")
> axis(side = 1, at = 2:8,
       labels = round(exp(2:8)))
> axis(side = 2, at = c(6, 6.5, 7, 7.5),
```

3 Die Höhe der Hinges (der Linien, die oben und unten an der Box angelegt werden) definiert sich durch den höchsten bzw. niedrigsten zulässigen Wert, der tatsächlich auftritt. Im Allgemeinen ist diese Grenze durch 1,5-mal die Höhe der Box (= Interquartilabstand) definiert.

4 Zusatzpakete werden in R über den Befehl library(<Paketname>) eingebunden. Diese Befehle finden Sie in den Code-Beispielen. Pakete müssen vor dem Einbinden installiert werden. Sollten diese Pakete noch nicht auf Ihrem Rechner installiert sein, können Sie das mit dem Befehl install.packages(„<Paketname>“) tun. Für das Paket „languageR“ lautet der Befehl somit install.packages(„languageR“).

```
       labels = round(exp(c(6, 6.5, 7, 7.5))))
> lines(lowess(lexdec$Frequency, lexdec$RT), lwd = 3)
> legend(x = "topright", col = c("red", "green"), pch = 19,
         legend = c("Tier", "Pflanze"), bty = "n",
         inset = 0.05, y.intersp = 1.5)
```

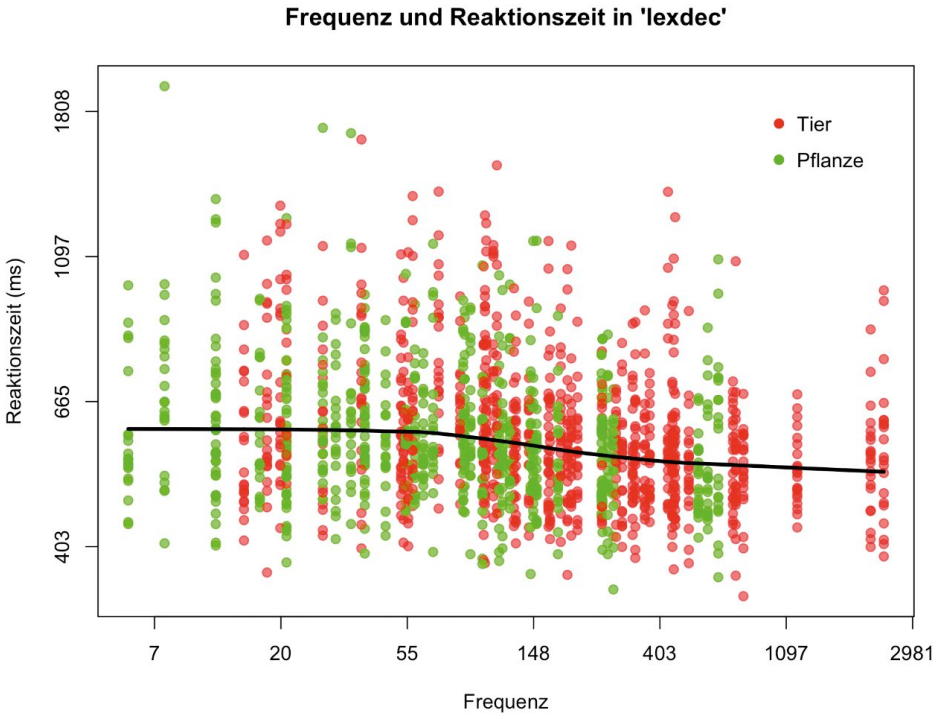

Abb. 1: Streudiagramm für die Reaktionszeit in Abhängigkeit der Häufigkeit im Datensatz „lexdec" mit Lowess-Anpassungskurve (schwarz) und Aufteilung nach semantischer Klasse.

Abb. 1 zeigt das Ergebnis nach Evaluation des abgedruckten Codes. Zunächst wird die zusätzliche Variable plot.col definiert, die je nach Ausprägung der Variable Class (semantische Klasse) einen Wert für ein transparentes Rot[5] für

5 Die Farben sind hier über RGB-Codes mit Alpha Channel angegeben. Die ersten zwei Werte definieren den Rotanteil, die zweiten zwei Ziffern den Grünanteil, die dritten zwei Ziffern den Blauanteil. Die Werte variieren zwischen oo (aus) und FF (komplett). Die letzten zwei Ziffern (Alpha Channel) definieren die Transparenz (variierend zwi-

Tierwörter („animal“) oder ein transparentes Grün für Pflanzenwörter („plant“) enthält. In der nächsten Zeile wird der eigentliche Plot erstellt, zunächst noch ohne Achsen (Parameter xaxt und yaxt beide „n“). Dann werden die beiden Achsen hinzugefügt, die anstatt der in der Variable enthaltenen logarithmierten Werte die zurücktransformierten Werte der Variablen enthalten (exp() ist die Umkehrung von log()). Als Nächstes wird über den Aufruf von lines() eine Lowess-Anpassungslinie (vgl. Cleveland 1981) dem augenblicklich verwendeten Grafikgerät hinzugefügt. Zuletzt wird die Legende konfiguriert und ebenfalls in den Plot eingefügt. Die verwendeten Parameter der Funktionen können jeweils über ?<Funktionsname> in R eingesehen werden (bspw. ?legend für die Hilfeseite von legend()).

Wenn wir versuchen, die Visualisierung aus Abb. 1 in das Schema in Tabelle 1 einzuordnen, so finden wir den vorliegenden Variablentyp in der vorletzten Zeile beschrieben. Zu visualisieren sind nämlich zwei kontinuierliche Variablen – Häufigkeit und Reaktionszeit (abgetragen auf x- und y-Achse) – sowie eine kategoriale Variable, die semantische Klasse (durch Farben symbolisiert).

Einige Zusatzpakete benutzen und erweitern die Basisfunktionalität von R. So ist das Paket „sciplot“ (Morales 2012) nützlich für die schnelle Exploration von Daten, die in einem faktoriellen Design[6] gesammelt wurden oder in einem solchen abbildbar sind. Mit den Funktionen lineplot.CI() und bargraph.CI() können Linien- oder Balkendiagramme mit Fehlerindikatoren erzeugt werden. Der Aufruf ist dabei recht einfach. Es muss eine (vorzugsweise kategoriale) Variable angegeben werden, die die x-Achse definiert sowie eine binäre oder kontinuierliche Variable, die die y-Achse definiert. Optional kann eine (kategoriale) Gruppierungsvariable angegeben werden. Die Berechnung der gruppenspezifischen Standardfehler übernimmt dann lineplot.CI() oder bargraph.CI(). Für die Exploration der eigenen Daten (sofern sie auf diese Weise abgebildet werden können) reicht das Ergebnis meist schon aus. Mit ein wenig Mehraufwand können auch publikationsreife Grafiken erstellt werden. Ein weiteres Beispiel aus dem Datensatz „lexdec“ verdeutlicht das Vorgehen. Wir tragen den Zusammenhang zwischen der Muttersprache der/des Befragten, der semantischen Klasse des Worts und der Reaktionszeit ab.

schen 00 = voll transparent und FF = voll deckend). Die Funktion alpha() aus dem Paket „scales“ erlaubt eine einfachere Definition von Transparenz in Kombination mit Farbnamen (bspw. alpha(„red“, 0.5) für halbtransparentes Rot).

6 Die Funktionen im Paket „sciplot“ eignen sich am besten dazu, den Zusammenhang zwischen zwei gekreuzten kategorialen unabhängigen Variablen und einer kontinuierlichen abhängigen Variable zu analysieren.

```
> library(sciplot)
> lineplot.CI(NativeLanguage, RT, Class, data = lexdec,
       xlab = "Muttersprache", ylab = "Reaktionszeit")
```

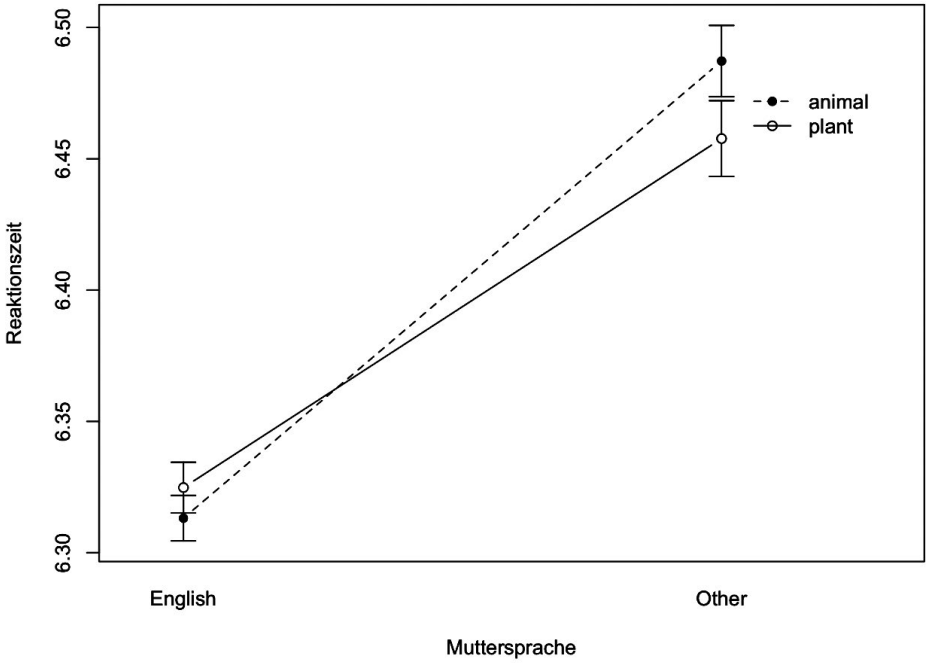

Abb. 2: Beispiel für einen Interaktionsplot mit dem Paket „sciplot“. Abgetragen ist die Beziehung zwischen der Muttersprache der Teilnehmenden, der semantischen Klasse des Worts und der Reaktionszeit in einer lexikalischen Entscheidungsaufgabe. Die Fehlerbalken symbolisieren 1 Standardfehler.

Abb. 2 zeigt das Ergebnis des Aufrufs. Die Grafik ist in dieser Form wohl noch nicht publikationsreif, insbesondere auch, weil die Linien zwischen den Datenpunkten zwar den potenziellen Interaktionseffekt (der noch anhand eines linearen Modells zu überprüfen wäre) gut hervorheben, im Grunde aber irreführend sind, da zwischen den Stufen „English“ und „Other“ des Faktors NativeLanguage keine Werte vorhanden sind. Für die Publikation wäre es daher eher angemessen, die Linien wegzulassen und die Datenpunkte leicht gegeneinander zu verschieben, damit die (Nicht-)Überlappung der Fehlerbalken leichter abgelesen werden kann.

Insbesondere für explanatorische Visualisierungen, die zur Veröffentlichung bestimmt sind, brauchen wir R-Grafiken in Bild-Dateien (.jpg/.png/...). Um ein solches Dateiformat zu erhalten, können wir entweder die Grafikexport-Funktion von RStudio[7] verwenden oder aber vor dem Erstellen der Grafiken mit den Funktionen jpeg(), png() usw. ein spezielles Grafikgerät starten. R wird dann den Grafikoutput nicht auf dem Bildschirm darstellen, sondern direkt in die Datei schreiben. Nachdem die Grafik in die Datei geschrieben wurde, müssen wir das Gerät wieder mit dev.off() schließen. Der Vorteil dieser Exportart liegt in den damit verbundenen Anpassungsmöglichkeiten. Über verschiedene Parameter wie bspw. height, width und res können wir die genauen Abmessungen und die Auflösung der zu erstellenden Grafik beeinflussen.[8] Der folgende Beispielaufruf verdeutlicht dieses Vorgehen:

```
> png("VisLing-testplot.png", width = 2000, height = 1600, res = 250)
> plot(rnorm(1000))
> dev.off()
```

Das Arbeitsverzeichnis (abrufbar mit der Funktion getwd()) enthält nun eine Datei „VisLing-tesplot.png“ mit einer Breite von 2000 Pixeln, einer Höhe von 1600 Pixeln und einer Auflösung von 250 ppi. Der Plot enthält 1000 zufällig normalverteilte Datenpunkte.

Sollen Grafikdateien in vektorbasierte Formate wie bspw. EPS (Encapsulated Postscript) oder SVG (Scalable Vector Graphics) exportiert werden, bietet R auch diese Möglichkeit. EPS-Dateien können über den Befehl postscript() erstellt werden, SVG-Dateien werden mit svg() erzeugt.

Aus den vorangegangenen Anwendungsbeispielen wurde deutlich, dass bereits das Basispaket in R äußerst vielfältige Visualisierungsmöglichkeiten bereithält. Selbstverständlich kann in diesem Rahmen nur ein kleiner Teil der Möglichkeiten präsentiert werden. Die Logik ist aber fast immer dieselbe: Eine Funktion beginnt entweder einen neuen Plot oder fügt Elemente in das augenblicklich geöffnete Grafikgerät ein. Auf diese Weise kann jeder Plot nach eigenen Wünschen konfiguriert und erweitert werden.

7 RStudio ist die wohl am weitesten verbreitete Entwicklungsumgebung für R. RStudio ist frei verfügbar unter www.rstudio.com/products/RStudio (letzter Zugriff am 13.05.2016). RStudio bietet viele Unterstützungen bei der Arbeit mit R an, auf die wir an dieser Stelle aber nicht umfassend eingehen können. Nützliche Funktionen sind unter anderem die Syntaxhervorhebung und -vervollständigung, viele Tastenkombinationen, die die Arbeit mit R unterstützen, Fehlerprüfungen und Versionskontrollfunktionen.

8 Für weitere Parameter siehe die Hilfeseite zu ?png.

2.2 Visualisierung kategorialer Daten: Das Paket „vcd"

Mit dem Paket „vcd" ist es möglich, kategoriale Daten auf verschiedene Art und Weise zu visualisieren. Unter kategorialen Merkmalen versteht man Größen, die endlich viele Ausprägungen besitzen und höchstens ordinalskaliert sind (Fahrmeir et al. 2007). Beispiele für kategoriale Variablen sind Muttersprache oder Wohnort eines Autors. Mithilfe des Pakets „vcd" sind eine Vielzahl an verschiedenen Plots möglich. Zu nennen sind hier beispielsweise der Mosaikplot, der Sieveplot[9] und der Assoziationsplot. Im Rahmen des vorliegenden Artikels werden wir zeigen, wie Mosaik- und Assoziationsplots erstellt und gelesen werden können. Hierzu nutzen wir ein Beispiel aus einer Analyse von Fürbacher (2015), die in einer Studie zu Genitivallomorphen im Deutschen bei starken Maskulina und Neutra zeigt, dass für die Variation der Genitivallomorphe *-es* und *-s* neben sprachimmanenten Faktoren auch regionale Einflüsse eine Rolle spielen können. Da kategoriale Daten üblicherweise durch Kontingenztabellen analysiert und visualisiert werden, bilden sie die Grundlage für Analysen mit dem Paket „vcd". Tabelle 2 zeigt eine solche Kontingenztabelle mit den absoluten Trefferzahlen für die Lemmata mit den Genitivendungen *-es* und *-s* in den Regionen Nordost, Südost (inklusive Österreich), Nordwest und Südwest (inklusive der Schweiz) aus der Analyse von Fürbacher (2015).[10]

Tabelle 2: Absolute Trefferzahlen für Lemmata mit den Genitivendungen *-es/-s* in Abhängigkeit von der Sprachregion nach Fürbacher (2015)

	-es	***-s***
Nordost	425	369
Südost	1435	573
Nordwest	239	235
Südwest	435	438

Wir erstellen in einem ersten Schritt die Kontingenztabelle in R. Hierzu generieren wir zunächst für jede Region einen Vektor mit den absoluten Häufigkeiten für die Genitivendungen *-es* und *-s*.

9 In Sieveplots werden Rechtecke geplottet, deren Flächeninhalte proportional zu den erwarteten Häufigkeiten der entsprechenden Tabellenzelle sind. In die Rechtecke werden Quadrate gezeichnet, die die beobachteten Häufigkeiten kennzeichnen. Die Dichte der ausgefüllten Rechtecke symbolisiert somit die Abweichung zwischen erwarteten und beobachteten Häufigkeiten.

10 Für eine detailliertere Beschreibung der Daten und Auswertungen der Studie vgl. Fürbacher (2015) und Hansen & Wolfer (i. Vorb).

```
> Nordost <- c(425, 369)
> Südost <- c(1435, 573)
> Nordwest <- c(239, 235)
> Südwest <- c(435, 438)
```

Über die Funktion rbind() werden die Vektoren zu einer Tabelle zusammengefügt und in der Variable tab gespeichert. Anschließend werden die Spaltennamen über die Funktion colnames() vergeben.

```
> tab <- rbind(Nordost, Südost, Nordwest, Südwest)
> colnames(tab) <- c("es", "s")
> tab
           es   s
Nordost   425 369
Südost   1435 573
Nordwest  239 235
Südwest   435 438
```

Auf Basis der vorliegenden Kontingenztabelle können nun die Analysen durchgeführt werden. Als Visualisierungsform wählen wir einen Mosaikplot (vgl. Abb. 3). Die Häufigkeiten werden über die Größe der Flächeninhalte der einzelnen Rechtecke dargestellt (vgl. Friendly 1994; Meyer et al. 2006). Darüber hinaus werden in dem Plot auch die signifikante Pearson-Residuen[11] (standardisierte Abweichungen der beobachteten von den erwarteten Häufigkeiten), angezeigt. Der Plot wird mit der Funktion mosaic() erstellt, die Schattierungen werden gemäß den Residuen über den Parameter shade = TRUE eingefügt.

```
> library(vcd)
> mosaic(tab, shade = TRUE)
```

Der Mosaikplot ist wie folgt zu lesen: Jedes Rechteck im Plot steht für eine Zelle der Kontingenztabelle. Der Flächeninhalt des Rechtecks links oben bezieht sich zum Beispiel auf die Häufigkeit der Genitivendung *-es* in der Region Nordost des deutschen Sprachraums. Es wird erkennbar, dass die meisten Tokens (*-es* und *-s*) im Korpus im Südosten vorkommen, denn die zweite Zeile an Rechtecken ist am höchsten. Die Einfärbung der Rechtecke symbolisiert die Pearson-Residuen. Sind die Pearson-Residuen signifikant (Betrag größer 1,97), werden sie im Plot eingefärbt: Die Farbe Blau steht für signifikante Abweichungen nach oben, die

11 Pearson-Residuen berechnen sich aus der Differenz der beobachteten und erwarteten Häufigkeiten geteilt durch die Wurzel der erwarteten Häufigkeiten.

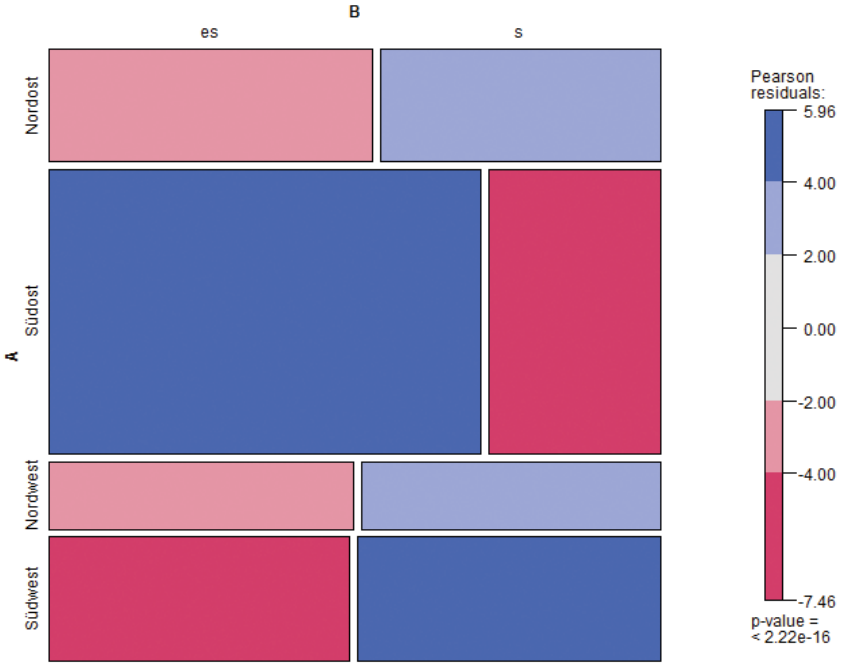

Abb. 3: Mosaikplot für die Genitivallomorphe *-es* und *-s* in Abhängigkeit von Regionen.

Farbe Rot für signifikante Abweichungen nach unten. Wird die Schwelle von +/- 4 überschritten, werden die Balken in einem stärkeren Ton eingefärbt, da hier von einer besonders starken Abweichung ausgegangen werden kann. Die Zuordnung von Einfärbungsgrad zu Residuenhöhe wird rechts neben dem Plot durch eine Legende dargestellt. Unterhalb der Legende wird außerdem der *p*-Wert eines Chi²-Tests ausgegeben, mit dem überprüft wird, ob die Ausprägungen der Variablen voneinander unabhängig sind (vgl. Bortz 2005: 168ff.).[12]

Bezogen auf das Beispiel zeigt der Mosaikplot, dass im Südosten die Genitivendung *-es* signifikant überrepräsentiert ist, während sie in den anderen Regionen signifikant unterrepräsentiert ist. Im Südwesten dominiert sogar die kürzere Endung *-s*.

Durch eine Anpassung über den Parameter labeling im Aufruf werden die genauen Werte für die standardisierten Residuen für jedes Rechteck im Plot angezeigt (vgl. Abb. 4):

12 Laut Konvention spricht man ab einer Irrtumswahrscheinlichkeit kleiner 5 % ($p < 0.05$) von einem signifikanten, ab 1% ($p < 0.01$) von einem hochsignifikanten und ab 0,1 % ($p < 0.001$) von einem höchstsignifikanten Unterschied.

```
> mosaic(tab, shade = TRUE,
       labeling=labeling_values(value_type=c("residuals")))
```

Eine weitere Möglichkeit, die standardisierten Abweichungen zwischen beobachteten und erwarteten Häufigkeiten darzustellen, bietet der Assoziationsplot (vgl. Cohen 1980; Friendly 1992; Meyer et al. 2006). Hierzu kann folgender Aufruf verwendet werden:

```
> assoc(tab, shade = TRUE,
       labeling = labeling_values(value_type=c("residuals")))
```

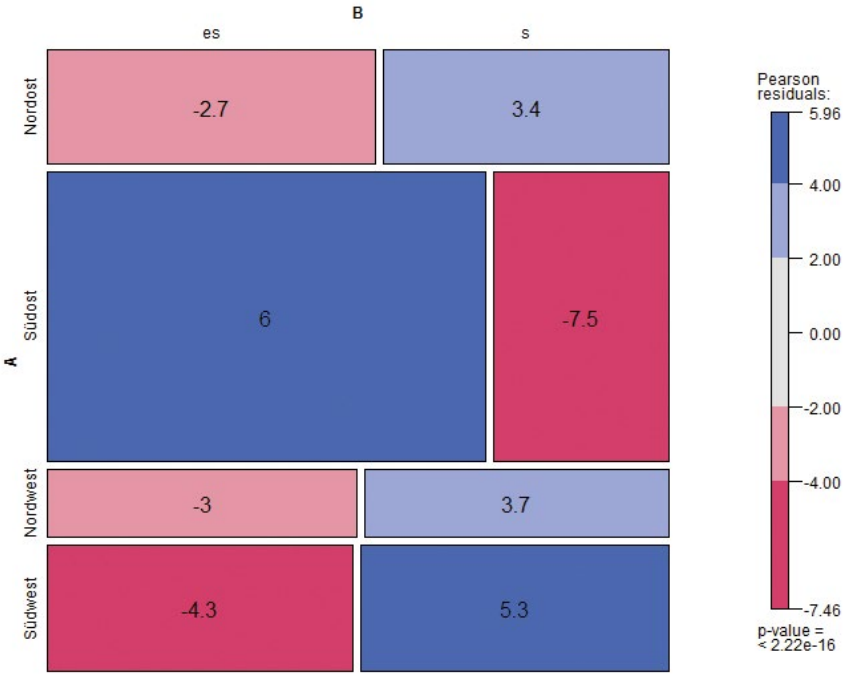

Abb. 4: Mosaikplot für die Genitivallomorphe *-es* und *-s* in Abhängigkeit von Regionen mit der Angabe der Pearson-Residuen für jedes Rechteck.

Abb. 5 zeigt den Assoziationsplot für das vorliegende Beispiel. Abgetragen werden die standardisierten Residuen der beiden Genitivendungen in Abhängigkeit von der regionalen Verteilung. Die Höhe der Balken entspricht der Höhe der

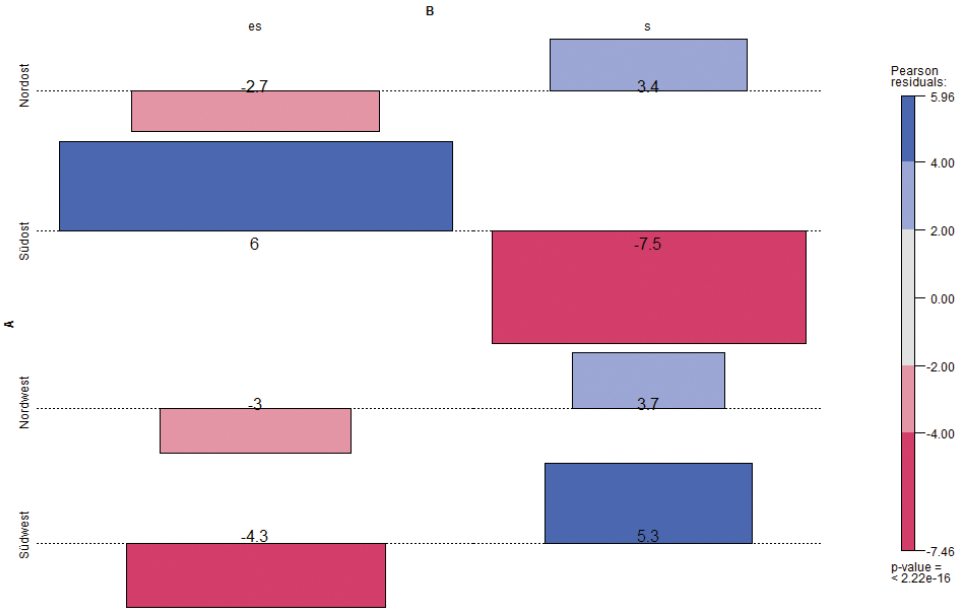

Abb. 5: Assoziationsplot für die Genitivallomorphe *-es* und *-s* in Abhängigkeit von Regionen mit Angabe der Pearson-Residuen für jedes Rechteck.

Abweichung: Balken oberhalb der gepunkteten Linie bedeuten höhere, Balken unterhalb der Linie niedrigere Werte als erwartet. Die genauen Werte der Pearson-Residuen werden hier ebenfalls durch die Anpassung des Parameters labeling angezeigt. Die Breite der Balken spiegelt die erwartete Häufigkeit der Realisierungsvarianten wider. Ist der Betrag des entsprechenden Pearson-Residuums größer 1,97, wird der Balken im Plot eingefärbt (vgl. Hansen-Morath et al., i. Vorb.).

Wie auch im Mosaikplot abzulesen ist, weist der Wert der Residuen in unserem Beispiel im Südosten mit 6,0 für *-es* bzw. -7,5 für *-s*, aber auch im Südwesten mit -4,3 für *-es* bzw. 5,3 für *-s* auf eine besondere Bedeutung der Abweichung für die Signifikanz der Unterschiede hin (vgl. Fürbacher 2015). Laut Fürbacher könnte der signifikant häufigere Gebrauch der *es*-Endung im Südosten durch eine kompensierende Funktion gegenüber der gesprochenen Sprache bedingt sein, da in dieser der Schwa-Laut gemieden wird (vgl. ebd.; Tatzreiter 1988: 79).

2.3 Visualisierung von Modellschätzern: Das Paket „effects“

In einigen Fällen ist es unerlässlich, nicht Rohdaten, sondern von statistischen Modellen *geschätzte* Daten zu visualisieren. Schon bei einfachen linearen Modellen[13], die mehr als einen Prädiktor enthalten, können die Schätzwerte von den Rohwerten abweichen. Unter Umständen kann sich das Verhältnis zweier Gruppen im Vergleich zwischen Modellschätzern und Zellmittelwerten sogar umkehren. Das kann vorkommen, da bei der Berechnung des Einflusses von Prädiktor x_1 Effekte anderer Prädiktoren x_2, x_3 ... x_n berücksichtigt werden und vice versa. Tragen wir also Zellmittelwerte ab, um die Ergebnisse eines linearen Modells zu visualisieren, haben wir das Problem, dass unsere Visualisierung nicht dieselben (und evtl. sogar in ihrer Aussage entgegengesetzte) Werte darstellt, wie wir im statistischen Modell berechnet haben.

Ein Minimalbeispiel anhand des Datensatzes „lexdec“ illustriert das Problem.

```
> library(languageR)
> library(effects)
> mod1 <- lm(RT ~ Class + Frequency, data = lexdec)
```

Mit der letzten Code-Zeile sagen wir die logarithmierte Reaktionszeit (RT)[14] aus der semantischen Klasse (Class, „plant“ vs. „animal“) und der Häufigkeit (Frequency) des Wortes mithilfe eines linearen Modells vorher. Obwohl die Interaktion der beiden Variablen nicht in der Vorhersage beachtet wurde, zeigt sich bereits, dass die Modellschätzwerte leicht von den Zellmittelwerten für Pflanzen und Tiere abweichen.

```
> tapply(lexdec$RT, lexdec$Class, mean) # Zellmittelwerte
  animal    plant
6.387746 6.381751
> Effect("Class", mod1) # Extraktion der Modellschätzer
Class effect
Class
  animal    plant
6.406457 6.358228
```

13 Mit linearen Modellen (bspw. einer linearen Regression) sagt man eine Kriteriumsvariable aus einer oder mehreren Prädiktorvariablen vorher (zur Einführung vgl. Bortz (2005: 483ff). In linearen Modellen können auch kategoriale Variablen als Prädiktoren verwendet werden. Im Beispiel in diesem Abschnitt werden eine kategoriale Variable (Class) und eine kontinuierliche Variable (Frequency) als Prädiktoren verwendet.

14 Reaktionszeiten sind typischerweise rechtsschief verteilt. Um diese Verteilung in eine Normalverteilung zu überführen, ist es zulässig, die Variable einer Log-Transformation zu unterziehen.

Der Unterschied zwischen beiden Gruppen ist bei den Schätzwerten größer: Die Differenz der geschätzten Mittelwerte beträgt dort 0,0482. Im Vergleich dazu beträgt die Differenz für die rohen Mittelwerte lediglich 0,00599. Durch die Einbeziehung der Häufigkeit als Prädiktor wird offenbar so viel Rauschen in den Daten kontrolliert, dass der Effekt des anderen Prädiktors (semantische Klasse) stärker hervortritt. Da wir das auch in der Visualisierung der Datenauswertung beachten wollen, sollten wir also nicht die Zellmittelwerte abtragen.

Durch eine Kombination von plot() mit den Funktionen des Pakets „effects" können wir die vom Modell geschätzten Werte visualisieren. Wir demonstrieren die Mächtigkeit des Pakets anhand eines Beispiels, in dem wir die Korrektheit der Antwort auf lexikalische Entscheidungsaufgaben vorhersagen. Da Korrektheit eine binäre Variable („correct" vs. „incorrect") ist, ist die logistische Regression (Pampel 2000) ein geeignetes statistisches Instrument. Eine solche fordern wir mit der Funktion glm() für „generalisiertes lineares Modell" und dem Parameter family = „binomial" an (vgl. Field 2012: 329f). Als Prädiktoren nehmen wir die Häufigkeit sowie die semantische Klasse des Wortes auf sowie die Interaktion zwischen den beiden Variablen. In den ersten beiden Code-Zeilen werden zunächst die Variablen Correct[15] und Frequency[16] vorbereitet.

```
> lexdec$Correct <- lexdec$Correct == "correct"
> lexdec$c.Frequency <- scale(lexdec$Frequency, scale = F)
> mod2 <- glm(Correct ~ c.Frequency * Class,
              data = lexdec, family = "binomial")
> plot(allEffects(mod2),
    multiline = T, ci.style = "bands",
    xlab = "Häufigkeit",
    ylab = "Geschätzte Wahrscheinlichkeit für korrekte Antwort",
    main = "Interaktionsplot Häufigkeit X Semantische Klasse")
```

Abb. 6 zeigt die grafische Ausgabe des letzten Aufrufs. Zuerst werden über die Funktion allEffects() alle Effekte höchster Ordnung[17] aus dem statistischen Modell mod2 extrahiert. Durch die Verschachtelung der Funktionen wird das

15 Da R die Stufen einer Variable alphabetisch sortiert, ist für die Variable Correct die erste Stufe „correct" und die zweite Stufe „incorrect". In der Vorhersage würde ein positives Vorzeichen somit bedeuten, dass sich die Wahrscheinlichkeit einer falschen Antwort erhöht. Da das nicht gerade intuitiv ist, drehen wir im ersten Aufruf die Stufen der Variable um („correct" wird zu TRUE, logisch wahr, und „incorrect" wird zu FALSE, logisch falsch).

16 Wann immer eine kontinuierliche Variable in eine Interaktion eingeht, sollte diese zentriert werden (Verschieben des Mittelwerts auf 0). Das geschieht mit dem zweiten Aufruf.

17 „Höchster Ordnung" bedeutet hier, dass bei interagierenden Variablen nur der Interaktionseffekt extrahiert wird, nicht die isolierten Effekte der Variablen. Würde das

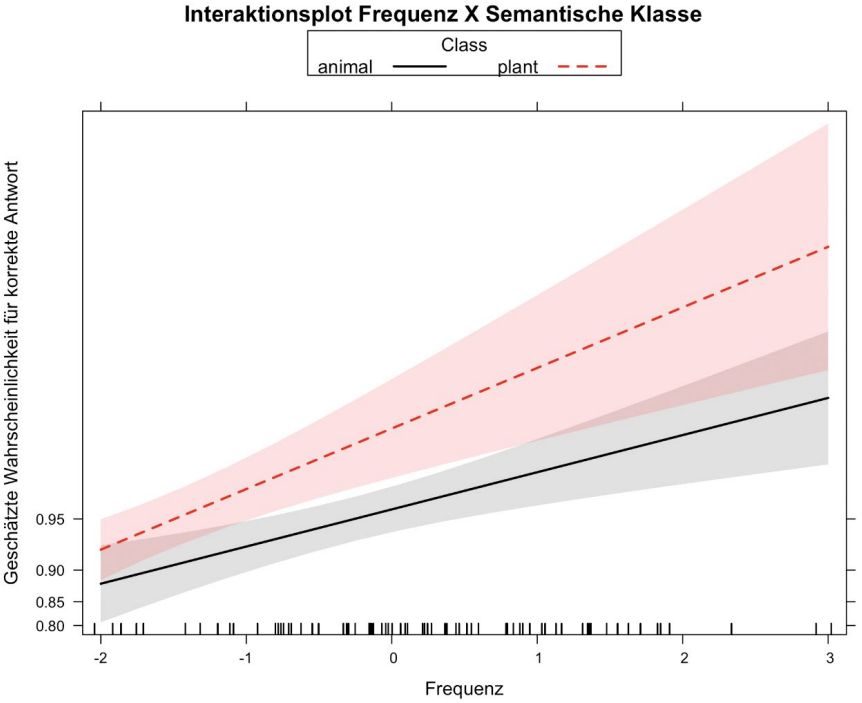

Abb. 6: Interaktionsplot für die geschätzte Wahrscheinlichkeit einer korrekten Antwort in einer lexikalischen Entscheidungsaufgabe in Abhängigkeit von Worthäufigkeit und semantischer Klasse. Fehlerindikatoren symbolisieren 95 %-Konfidenzintervalle.

Ergebnis dieser Extraktion direkt an plot() übergeben.[18] Dieser Aufruf enthält einige zusätzliche Argumente. xlab, ylab und main steuern lediglich die Achsen- und Titelbeschriftungen. multiline = T sorgt dafür, dass beide Linien für die Stufen der interagierenden Variable Class in einem Plot abgetragen werden und nicht – wie es in der Standardeinstellung der Fall wäre – in zwei separaten Panels. ci.style = „bands“ bewirkt, dass um diese Linien Konfidenzintervalle gezeichnet werden. Auffällig ist die Beschriftung der y-Achse, auf der die tatsächlich geschätzten Wahrscheinlichkeiten abgetragen werden. Durch die Transformationen der Kriteriumsvariable, die einer logistischen Regression inhärent

Modell noch einen dritten Prädiktor enthalten, der nicht in eine Interaktion eingeht, würde dieser ebenfalls extrahiert.

18 Intern erkennt R, dass es sich bei dem an plot() übergebenen Objekt um ein effects-Objekt handelt, und benutzt daher plot.eff(). Die Hilfeseite zu dieser Funktion (?plot.eff) enthält Erklärungen zu den übergebenen und einigen anderen Parametern.

sind, müssen die Abstände zwischen den Skalenstrichen, die die geschätzte Wahrscheinlichkeit angeben, entsprechend angepasst werden. Diese Anpassung sowie die Rücktransformation in einfacher interpretierbare Wahrscheinlichkeiten werden vom Zusatzpaket „effects“ übernommen.

Die Interaktion der beiden Variablen Class und c.Frequency ist nicht signifikant. Dies kann man über einen Aufruf von summary(mod2) sehen: Die Werte für die Interaktion lauten $\beta = 0{,}342$; $SE = 0{,}242$; $z = 1{,}413$ und $p = 0{,}158$. Der Unterschied der beiden Geraden-Steigungen in Abb. 6 ist also nicht groß genug, um anzunehmen, dass die Häufigkeit bei Tiernamen einen anderen Einfluss auf die Korrektheit hat als bei Pflanzennamen.

Abb. 6 enthält außerdem einen sogenannten „rug plot“. Die kleinen Striche am unteren Rand des Plotbereichs geben an, an welchen Stellen auf der x-Achse (also bei welchen Häufigkeiten) sich tatsächlich Datenpunkte befinden. Daraus gewinnen wir einerseits einen Eindruck von der Häufigkeitsverteilung der im Experiment verwendeten Wörter. Andererseits hilft es uns einzuschätzen, ob dem statistischen Modell über das komplette Spektrum hinweg ausreichend Datenpunkte zugrunde lagen, um eine Schätzung vorzunehmen. Würde sich im Rug-Plot beispielsweise zeigen, dass nur sehr seltene und sehr häufige Wörter in der Datenbasis vorhanden sind, könnte das auf ein Problem hindeuten. Im Falle von Abb. 6 scheinen über das komplette Häufigkeitsspektrum hinweg Datenpunkte vorhanden zu sein.

Wir fassen also für diesen Abschnitt zusammen: Wollen wir die Ergebnisse eines statistischen Modells (bspw. eines linearen Modells) visualisieren, sollten wir nicht auf Rohmittelwerte zurückgreifen, sondern die Schätzwerte aus dem Modell selbst extrahieren. Verschiebungen, die sich durch die Aufnahme anderer Prädiktoren ergeben, werden so auch in die Visualisierung übernommen. Das Paket „effects“ stellt Funktionen bereit, die die Extraktion und Visualisierung der Modellschätzer übernehmen. Dies ist unter anderem auch für die komputational aufwändigeren gemischten Modelle der Fall. Die Plots sind auf vielfältige Weise über Parameter anpassbar. Wenn für eine Veröffentlichung ein ganz bestimmtes Format an Schaubildern nötig ist, kann man auch die extrahierten Werte in eigens erstellte Plots übernehmen.

3. Alternative Plot-Pakete

Mitte Mai 2016 waren allein auf dem Comprehensive R Archive and Network[19] (CRAN) ca. 8300 Zusatzpakete für R verfügbar. Darin sind Pakete aus anderen Repositorien wie bspw. Bioconductor[20] noch nicht mit eingerechnet. Unter dieser großen Menge befinden sich natürlich auch einige Pakete, die speziell auf die Erzeugung von Grafiken, Schaubildern und Visualisierungen ausgerichtet sind. Der CRAN Task View zu „Graphics"[21] ist ein sehr guter Ausgangspunkt, um sich einen Überblick zu verschaffen. Wer sich für die Visualisierung von räumlichen Daten (bspw. auf Landkarten) interessiert, findet außerdem ausführliche Informationen auf dem Task View zu „Spatial Data".[22] In diesem Abschnitt werden wir uns auf die Vorstellung zweier Pakete beschränken. Das erste, „ggplot2", ist relativ prominent und breit genutzt. Das zweite, „rCharts", befindet sich noch in der Entwicklungsphase und wird noch nicht von vergleichbar vielen Personen benutzt.[23] Es scheint uns aber insofern interessant zu sein, als es eine Verbindung herstellt zwischen R und der JavaScript-Visualisierungsbibliothek d3.js, „Data Driven Documents"[24], die in jüngster Zeit an Bedeutung gewonnen hat.

3.1 „Grammar of Graphics": Das Paket „ggplot2"

Hadley Wickham erklärt die Grundidee jeglicher Grafikerstellung folgendermaßen: „[A] statistical graphic is a *mapping* from *data* to *aesthetic attributes* (colour, shape, size) of *geometric objects* (points, lines, bars)" [Hervorhebungen SW und SHW] (Wickham 2009: 3). In diesem Zitat sind alle relevanten Konzepte, die in „ggplot2" zum Einsatz kommen, bereits angesprochen. Die *Daten* setzen sich aus unseren Beobachtungen, den Variablen im Datensatz sowie deren Merkmalsausprägungen zusammen. Über den Abbildungsprozess *(mapping)* werden diese Daten dann auf die ästhetischen Attribute der *geometrischen Objekte* (in „ggplot2"-Terminologie „geoms") abgebildet. Wie Eugster und Scheipl in einer Präsentation[25] anmerken, ist die *Grammar of Graphics* von Wilkinson (2005)

19 https://cran.r-project.org/web/packages (letzter Zugriff am 13. Mai 2016).
20 https://www.bioconductor.org (letzter Zugriff am 13. Mai 2016).
21 https://cran.r-project.org/web/views/Graphics.html (letzter Zugriff am 13. Mai 2016).
22 https://cran.r-project.org/web/views/Spatial.html (letzter Zugriff am 13. Mai 2016).
23 Dies sind unsere Einschätzungen. Die tatsächliche Nutzung eines Pakets lässt sich nur schwer operationalisieren. Insbesondere wenn das Paket nicht auf CRAN verzeichnet ist, wie im Falle von „rCharts".
24 https://d3js.org/ (letzter Zugriff am 13. Mai 2016).
25 http://www.statistik.lmu.de/~scheipl/downloads/grafiken-05-ggplot2.pdf (letzter Zugriff am 13. Mai 2015).

keine „Anleitung, welche Grafik zu erzeugen ist, um eine konkrete Fragestellung zu untersuchen“, und keine „Spezifikation, wie eine statistische Grafik ausschauen sollte“. Vielmehr ist diese Grammatik als ein „formales Regelwerk“ zu verstehen, „welches die Zusammenhänge zwischen allen Elementen einer (gängigen) statistischen Grafik beschreibt“ (siehe Folie 6 der Präsentation von Eugster und Scheipl).

Die Funktionsweise von „ggplot2“ soll anhand eines einfachen Beispiels erläutert werden. Wir benutzen dazu den Datensatz „ratings“ aus dem Paket „languageR“, der subjektive Einschätzungen (z. B. das Gewicht) zu 81 Items (bspw. *ant*, *apple*, *woodpecker*) enthält (für nähere Erläuterungen siehe ?ratings).

```
> library(languageR)
> library(ggplot2)
> ggplot(data = ratings) +
        aes(x = Frequency, y = meanWeightRating, col = Class) +
        geom_point() +
        geom_smooth() +
        geom_rug()
```

Ergebnis des Aufrufs ist die in Abb. 7 abgedruckte Grafik. Man erkennt im Aufruf die verschiedenen Elemente des obigen Zitats wieder: In der ersten Zeile wird der Datensatz spezifiziert. In der Funktion aes() werden die ästhetischen Abbildungen vorgenommen. Dabei wird die x-Achse mit der Korpushäufigkeit, die y-Achse mit der durchschnittlichen Einschätzung des Gewichts und die Farbe der Datenpunkte mit der semantischen Klasse („animal“ vs. „plant“) belegt. Zuletzt werden noch drei geeignete geoms hinzugefügt. Die Datenpunkte selbst mit geom_point(), die Anpassungslinien mit geom_smooth() sowie die Rug-Plots mit geom_rug(). Zur Interpretation von Rug-Plots, siehe Abschnitt 2.3. Die verschiedenen Elemente werden mit dem Zeichen + verknüpft[26].

„ggplot2“ übernimmt einige Dinge selbst: Die Legende wird automatisch erzeugt und auch die Anpassungslinien werden automatisch für die beiden farblich unterschiedenen Gruppen getrennt erzeugt. Würden wir nicht nach Farbe gruppieren, wäre nur eine Anpassungslinie für die komplette Punktwolke zu sehen. Die geoms übernehmen die jeweils geeigneten Zuordnungen aus dem Aufruf von aes() – deshalb sind auch die Linien der Rug-Plots nach Gruppen eingefärbt. So kann man relativ leicht erkennen, dass sich die beiden Gruppen auf der y-Achse kaum mischen. Das Gewicht von Tieren wird offenbar systematisch als höher eingeschätzt als das von Pflanzen/Früchten. Die Anpassungslinien

26 Die Bedeutung von + ist hier nicht als arithmetische Operation (Addition) zu verstehen und auch nicht im Sinne eines Formeloperators wie im lm()-Aufruf in Abschnitt 2.3.

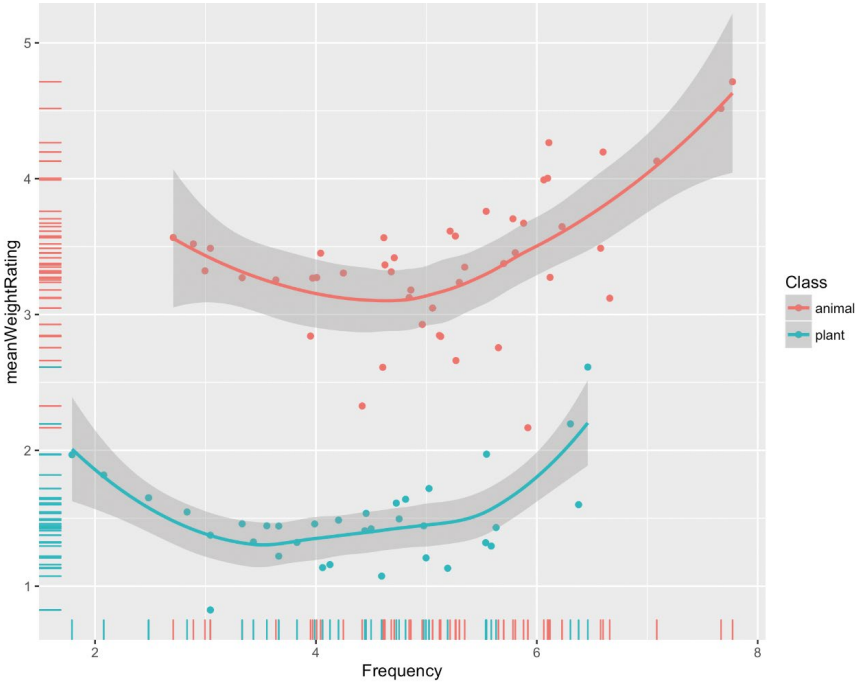

Abb. 7: Mit dem Paket „ggplot2“ erstellte Grafik, die den Zusammenhang zwischen der Häufigkeit eines Wortes, dessen semantischer Klasse und der durchschnittlichen Einschätzung des Gewichts des bezeichneten Tieres bzw. der Pflanze visualisiert. Eingefügt wurden außerdem Anpassungslinien für beide Gruppen sowie Rug-Plots für die x- und y-Werte.

deuten auf einen nicht-linearen, u-förmigen Zusammenhang zwischen Worthäufigkeit und durchschnittlicher Schätzung des Gewichts hin, was natürlich noch statistisch abzusichern wäre.

Grafiken, die mit „ggplot2“ erstellt wurden, sehen meist schon in der Standardausführung sehr ansprechend und „aufgeräumt“ aus und sind insbesondere zu explanatorischen Zwecken gut geeignet, bspw. zur Verwendung in Veröffentlichungen. Eine Grafik wie Abb. 5 mit dem Basisgrafikpaket von R zu erstellen, ist zwar nicht unmöglich, es bräuchte aber sicherlich deutlich mehr Code-Zeilen als mit der Erstellung mit „ggplot2“.

Wir konnten hier nur einen kleinen Eindruck vom Paket „ggplot2“ vermitteln und versuchen, die Grundidee zu verdeutlichen. Es gibt eine Reihe sehr guter Online-Tutorials für „ggplot2“. Empfohlen sei hier ein ausführliches Tutorial, das vom Institute for Quantitative Social Science der Harvard University

bereitgestellt wird[27]. Außerdem bietet die mit dem Paket assoziierte Dokumentationsseite[28] u. a. einen Überblick über alle geoms, die von dem Paket unterstützt werden.

3.2 Interaktive Grafiken mit d3: Das Paket „rCharts“

Ein weiteres Paket, das wohl eher explanatorisch als explorativ ausgerichtet ist, ist das Paket „rCharts“[29]. Es scheint geeignet zu sein, in Zukunft die Lücke zwischen R und der JavaScript-Bibliothek Data Driven Documents (d3) schließen zu können. Interessant ist d3 unter anderem aufgrund der Möglichkeit, interaktive Grafiken zu erstellen. „rCharts“ ist momentan noch nicht über den zentralen R-Paketverteiler CRAN verfügbar, sondern nur über GitHub.[30] Die Installation ist nicht sonderlich schwierig, wenn man zuerst das R-Paket „devtools“ installiert. „devtools“ enthält eine Funktion, mit dem man von GitHub R-Pakete installieren kann.[31] Mit der folgenden Sequenz von Befehlen sollte „rCharts“ auf dem eigenen Rechner verfügbar sein:

```
> install.packages("devtools")
> library(devtools)
> install_github('rCharts', 'ramnathv')
> library(rCharts)
```

Ein warnendes Wort vorweg: Mit „rCharts“ können zwar ansprechende interaktive Grafiken erstellt werden, da sich das Paket noch in der Entwicklung befindet, ist die Dokumentation jedoch noch recht lückenhaft. Das heißt, dass man entweder grundlegende Kenntnisse in JavaScript besitzen muss, um mit dem Paket gute Ergebnisse zu erzielen, oder etwas experimentieren muss, bis man zum gewünschten Ergebnis kommt. Wer es aber schließlich geschafft hat, einen Plot zu erstellen, wird mit einem visuell ansprechenden interaktiven Plot „belohnt“. Die Ergebnisse der folgenden Aufrufe sind lediglich über Links verfügbar, da „rCharts“ HTML-Dateien erstellt, deren interaktive Elemente im gedruckten

27 http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html (letzter Zugriff am 13. Mai 2016).

28 docs.ggplot2.org (letzter Zugriff am 13. Mai 2016).

29 Die mit dem Paket assoziierte Website ist unter rcharts.io (letzter Zugriff am 13. Mai 2016) erreichbar.

30 GitHub ist ein Webhosting-Service, auf dem man u. a. Code für R-Pakete bereitstellen und versionieren kann.

31 Da sich, wie erwähnt, „rCharts“ noch in der aktiven Entwicklung befindet, kann sich die Installationsprozedur eventuell ändern.

Text selbstverständlich nicht dargestellt werden können. Wir werden mit einem Beispiel beginnen, mit dem wir das in Abschnitt 3.1 in Abb. 7 gezeigte Diagramm replizieren werden.

```
> library(rCharts)
> library(languageR)
> pl <- nPlot(x = "Frequency", y = "meanWeightRating", group = "Class",
data = ratings, type = "scatterChart")
> pl$params$width <- 1000
> pl$params$height <- 600
> pl$yAxis(axisLabel = 'Mittleres geschätztes Gewicht')
> pl$xAxis(axisLabel = 'Häufigkeit')
> pl$chart(showDistY = 'true')
> pl$chart(showDistX = 'true')
> pl$chart(color = c("orange", "blue"))
> pl$chart(tooltipContent = "#! function (a, b, c, data) {
    return data.point.Word
           } !#")
> pl
```

Das Ergebnis dieses Aufrufs ist eine HTML-Datei, die ein interaktives Diagramm enthält und unter https://doi.org/10.11588/data/6SO6TG verfügbar ist (Datei „pl.html"). Was genau geschieht im obigen Aufruf? Zunächst laden wir die nötigen Pakete. Dann wird ein Plot mit der Funktion nplot() erstellt. Die Funktion nplot() nutzt die d3-Bibliothek „NVD3"[32] und kann mehrere Arten von Visualisierungen erstellen. Wir wählen hier den Typ „scatterChart". Außerdem müssen wir die Variablen für x- und y-Achse sowie den Datensatz angeben. Die Gruppierungsvariable ist fakultativ und kontrolliert hier u. a. die Farbe der Datenpunkte. Im Folgenden werden einige Eigenschaften des in der Variable „pl" gespeicherten Plots verändert: zuerst die Abmessungen (width, height), dann die Beschriftungen der Achsen (axisLabel). Über die beiden nächsten Zeilen (showDistX/Y) aktivieren wir die Option, dass die Verteilung der Datenpunkte an der x- und y-Achse über Rug-Plots dargestellt wird. Die Kolorierung der Datenpunkte für die beiden semantischen Klassen wird in der nächsten Zeile in Orange und Blau geändert. Die vorletzte Zeile enthält einen Aufruf, der den Inhalt des Tooltips steuert. Dieser wird angezeigt, wenn die Benutzerin/der Benutzer mit der Maus über einen Datenpunkt fährt. In diesem Fall soll das konkrete Wort angezeigt werden. Diese Zeile enthält schon ein wenig JavaScript-Code. In der letzten Zeile wird der Plot nur noch „ausgeführt" und im HTML-Viewer von RStudio angezeigt.

32 http://nvd3.org/ (letzter Zugriff am 17. Mai 2016).

Wo liegen nun die interaktiven Elemente dieser Visualisierung? Zunächst können oben links mit einem Klick auf den Farbpunkt neben „plant“ und „animal“ die Gruppen aktiviert und deaktiviert werden. Deaktiviert man eine der beiden Gruppen, verschwinden die Datenpunkte und die Skalierung der Achsen wird neu auf die dargestellten Datenpunkte ausgerichtet. Über die Aktivierung der „Magnify“-Option oben links wird in einen „Fischaugenmodus“ umgeschaltet, in dem über die Maus gesteuert werden kann, welche Bereiche des Graphen vergrößert dargestellt werden. Die Achsen werden hier dynamisch skaliert.

Einen weiteren Mehrwert der interaktiven Visualisierung erkennt man, wenn man mit der Maus über einzelne Datenpunkte fährt. Hier geschehen mehrere Dinge. Erstens wird ein Tooltip eingeblendet, in dem das tatsächlich bewertete Wort dargestellt wird. Das ist insbesondere hinsichtlich der Menge der im Schaubild dargestellten Informationen interessant. Man stelle sich eine statische Visualisierung vor, in der zu jedem Datenpunkt das Wort annotiert ist (auch das ist selbstverständlich in R möglich). Das Schaubild droht dann recht schnell unübersichtlich zu werden. Durch das interaktive Element des Tooltips kann die/der Betrachtende selektiv diese Information anfordern.

Fährt man mit der Maus über einzelne Datenpunkte, werden außerdem senkrechte und horizontale Geraden zu den Achsen gezeichnet und die Werte des Punktes auf den Achsen dargestellt. So lässt sich für jeden Punkt der genaue Wert ablesen. Alle interaktiven Elemente werden mit einer Übergangsanimation dargestellt, was es visuell etwas ansprechender macht.

Ein zweites Beispiel soll eine Visualisierungsart zeigen, die in diesem Kapitel zwar schon angesprochen, aber noch nicht anhand eines Beispiels illustriert wurde: ein gruppiertes Balkendiagramm. Wir bedienen uns dazu des Beispieldatensatzes „dative“ aus dem Paket „languageR“. Dieser Datensatz enthält Informationen zu 3263 Realisierungen der englischen „double object“-Konstruktion, in der das Phänomen der „dative alternation“ wirkt („John gives Mary a book“ vs. „John gives the book to Mary“). Im ersten Fall ist der Rezipient der Geben-Handlung, Mary, als eine Nominalphrase (NP) realisiert, im zweiten Fall als eine Präpositionalphrase (to Mary, PP). Wir wollen die Verteilung der Fälle in diesem Datensatz bezüglich der semantischen Klasse des Verbs visualisieren. Diese ist 5-stufig codiert: a = abstract („give it some thought“ / „give some thought to it“), c = communication („tell me your name“ / „tell your name to me“), f = future transfer of possession („promise her some money“ / „promise some money to her“), p = prevention of possession („deny him the book“ / „deny the book to him“) und t = transfer of possession („give him a ring“ / „give the ring to him“). Wir visualisieren also zwei kategoriale Variablen in ihrem Zusammenhang: die Realisierung des Rezipienten (NP vs. PP) und die semantische Klasse des Verbs (a, c, f, p oder t) und interessieren uns für die Häufigkeit, mit der jede der

Kombinationen auftritt. Wir codieren zunächst die Variable SemanticClass um, um die Interpretierbarkeit des Schaubilds zu erleichtern.

```
> levels(dative$SemanticClass) <- c("abstract", "communication",
                "future transfer",
                "prevention of transfer", "transfer")
> table(dative$SemanticClass, dative$RealizationOfRecipient)
                            NP   PP
  abstract                1176  257
  communication            355   50
  future transfer           47   12
  prevention of transfer   225    3
  transfer                 611  527
```

Wir kennen nun schon die Häufigkeitsverteilung, die uns interessiert, und wollen diese jetzt mit rCharts interaktiv visualisieren.

```
> dative2 <- as.data.frame(table(dative$SemanticClass,
                      dative$RealizationOfRecipient))
> names(dative2) <- c("SemanticClass", "RealizationOfRecipient",
                "Freq")
> pl2 <- nPlot(Freq ~ RealizationOfRecipient,
               group = "SemanticClass", data = dative2,
               type = "multiBarChart")
> pl2$yAxis(tickFormat = "#! d3.format(',.0f')!#")
> pl2
```

In der ersten Zeile wird zunächst eine Datentabelle (ein „Dataframe“) erstellt, mit dem das Paket „rCharts“ umgehen kann. Das ist lediglich eine andere Darstellungsform der oben abgedruckten Kontingenztabelle. In diesem Dataframe legen wir zunächst geeignete Spaltenüberschriften mit der Funktion names() fest, dann folgt der eigentliche Plotaufruf, in dem wir die Häufigkeit in Abhängigkeit der Realisierung des Rezipienten gruppiert nach der semantischen Klasse abtragen. Der Typ des Plots ist in diesem Fall ein multiBarChart. Der Plot wird zunächst in der Variable pl2 gespeichert. Im letzten Schritt passen wir das Format der y-Achse so an, dass keine Nachkommastellen angezeigt werden, denn diese sind bei ganzzahligen Häufigkeiten unnötig und verwirren eher. Mit der letzten Zeile wird der Plot schließlich „ausgeführt“.

Das Ergebnis des Plotaufrufs ist wiederum eine HTML-Datei, die das interaktive Schaubild enthält und unter https://doi.org/10.11588/data/6SO6TG abrufbar ist (Datei „pl2.html“). Zunächst ist kein bedeutender Unterschied zu statischen

Balkendiagrammen feststellbar. Doch auch hier bieten die interaktiven Elemente einen Mehrwert: Wiederum kann man rechts oben in der Farb-Legende einzelne Gruppen an- und ausschalten, was gegebenenfalls zu einer Neuskalierung der y-Achse führt. Wenn man mit der Maus über einzelne Balken fährt, wird außerdem ein Tooltip angezeigt, der die genaue Anzahl der Fälle sowie die Kategorie enthält (bspw. „communication – 355 on NP"). Besonders hervorzuheben ist die Möglichkeit, von einem gruppierten auf ein gestapeltes Balkendiagramm umzuschalten. Hierzu muss lediglich oben links „stacked" statt „grouped" aktiviert werden. In der Visualisierung der „dative alternation"-Daten lässt sich mit einem gestapelten Balkendiagramm beispielsweise die Gesamtzahl der Fälle NP vs. PP einfacher vergleichen. Ein gruppiertes Balkendiagramm eignet sich dagegen besser für den Vergleich einzelner Gruppen.

Mithilfe des Pakets „rCharts" können wir noch weitere auf NVD3 basierte Grafiken direkt in R erstellen. Hierzu gehören gestapelte Flächendiagramme (stackedAreaChart), horizontale Balkendiagramme (multiBarHorizontalChart) sowie Tortendiagramme (pieChart) und Liniendiagramme ohne (lineChart) und mit einer speziellen Auswahlfunktion (lineWithFocusChart). Außerdem können noch andere auf d3 basierte Bibliotheken angesprochen werden (bspw. Polychart, Morris, xCharts und HighCharts). Einen relativ umfassenden Überblick über die Fähigkeiten von „rCharts" bietet die Projekthomepage[33] sowie diverse web-basierte Tutorials[34].

4. Schluss

Wir haben einen kleinen Ausschnitt der Visualisierungsmöglichkeiten gezeigt, die R bietet – sowohl mit explorativer als auch mit explanatorischer Ausrichtung. Unseren Visualisierungen lagen statistische Berechnungen zugrunde, die mit einer unterschiedlichen Anzahl und Art von Variablen arbeiteten. Überlegt man sich bereits im Vorfeld der Grafikerstellung, wie viele und welche Variablen visualisiert werden sollen, fällt die Wahl einer geeigneten Visualisierungsart meist deutlich einfacher.

Mit der Basisausstattung von R stehen den Benutzer/innen schon viele Visualisierungsmöglichkeiten zur Verfügung. Für Spezialaufgaben sind bestimmte Pakete jedoch besser geeignet, da sie der Benutzerin/dem Benutzer unter Umständen eine Menge Kodierungs- oder Programmierarbeit abnehmen. Wir konnten

33 http://ramnathv.github.io/rCharts (letzter Zugriff am 19. Mai 2016).

34 Unter http://www.rpubs.com/dnchari/rcharts (letzter Zugriff am 20. Mai 2016) ist eine Sammlung von Programmcode für einige Schaubilder verfügbar, die mit rCharts erstellt werden können.

hier nur einen kleinen Ausschnitt der verfügbaren Pakete vorstellen, haben aber versucht, eine Auswahl zu treffen, die in vielen Bereichen der linguistischen Forschung relevant sein können. Dabei ist die Visualisierung kategorialer Daten besonders interessant (Paket „vcd“). Aber auch die Möglichkeit, mit statistischen Modellen gewonnene Schätzer auf einfache Weise zu extrahieren und ohne Mehraufwand zu visualisieren, ist äußerst relevant – spätestens in Veröffentlichungen, in denen die Ergebnisse der statistischen Modelle und nicht einfach die Rohwerte dargestellt werden sollen (Paket „effects“). Zuletzt haben wir zwei Pakete vorgestellt, die als komplette Umgebungen für die Erstellung von Grafiken gesehen werden können und somit das Portfolio von R als mächtiges Visualisierungswerkzeug erweitern. Das Paket „ggplot2“ tritt dabei mit dem Anspruch an, eine „Grammar of Graphics“ zu implementieren. Das noch in der Entwicklung befindliche Paket „rCharts“ schließt die Lücke zur JavaScript-Bibliothek d3 und verknüpft R somit mit einer anderen mächtigen Visualisierungsumgebung.

5. Bibliografie

Baayen, R. Harald. 2011. languageR: “Data sets and functions with ‘Analyzing linguistic data: A practical introduction to statistics’” [Computer software manual]. Abgerufen von http://CRAN.R-project.org/package=languageR (R package version 1.4)

Bortz, Jürgen. 2005. *Statistik für Human- und Sozialwissenschaftler.* 6. Aufl. Heidelberg: Springer.

Cohen, Ayala. 1980. “On the graphical display of the significant components in a two-way contingency table.” *Communications in Statistics. Theory and Methods* 9: 1025–1041.

Cleveland, William S. 1981. “LOWESS: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician* 35: 54.

Fahrmeir, Ludwig, Rita Künstler, Iris Pigeot und Gerhard Tutz. 2007. *Statistik: Der Weg zur Datenanalyse.* Berlin: Springer.

Field, Andy, Jeremy Miles und Zoë Field. 2012. *Discovering statistics using R.* Los Angeles: Sage.

Fox, John .2003. “Effect displays in R for Generalised Linear Models.” *Journal of Statistical Software* 8: 1-24.

Friendly, Michael. 1992. “Graphical methods for categorical data. SAS User Group International Conference Proceedings” 17: 190–200. http://www.math.yorku.ca/SCS/sugi/sugi17-paper.html (letzter Zugriff am 19.05.2016).

Friendly, Michael. 1994. “Mosaic displays for multi-way contingency tables.” *Journal of the American Statistical Association* 89: 190–200.

Fürbacher, Monica. 2015. "Variation zur starken Genitivmarkierung. Spezialstudie: Regionale Verteilung." https://dgd.ids-mannheim.de/korpusgrammatik/5087 (letzter Zugriff am 19. Mai 2016.).

Hansen, Sandra und Sascha Wolfer. 2016. „Standardisierte statistische Auswertung von Korpusdaten im Projekt ‚Korpusgrammatik' (KoGra-R)". In: *Grammatische Variation. Empirische Zugänge und theoretische Modellierung,* herausgegeben von Marek Konopka und Angelika Wöllstein. Berlin: de Gruyter, 345–356.

Hansen-Morath, Sandra, Roman Schneider, Hans-Christian Schmitz und Sascha Wolfer. In Vorbereitung. „KoGra-R: Standardisierte statistische Auswertung von Korpusrecherchen." In: *Grammatik im Korpus,* herausgegeben von Eric Fuß, Marek Konopka und Angelika Wöllstein.

Meyer, David, Achim Zeileis und Kurt Hornik. 2007. „The strucplot framework: Visualizing multi-way contingency tables with vcd." *Journal of Statistical Software* 17 (3): 1–48. http://www.jstatsoft.org/v17/i03/ und verfügbar als Vignette („strucplot", package = „vcd").

Meyer, D., A. Zeileis, A. und K. Hornik. 2015. "vcd: Visualizing categorical data" [Computer software manual]. (R package version 1.4-1).

Morales, M. und R Core Team. 2012. "sciplot: Scientific graphing functions for factorial designs" [Computer software manual]. Abgerufen von https://CRAN.R-project.org/package=sciplot (R package version 1.1-0).

Pampel, Fred C. 2000. *Logistic regression: A primer.* London: Sage.

R Core Team 2016. "R: A language and environment for statistical computing" [Computer software manual]. Vienna, Austria. Abgerufen von http://www.R-project.org/

Tatzreiter, Herbert. 1988. „Besonderheiten der Morphologie in der deutschen Sprache in Österreich." In: *Das österreichische Deutsch,* herausgegeben von Peter Wiesinger. Wien: Böhlau, 1988, 76.

Vaidyanathan, Ramnath. 2013. "rCharts: Interactive charts using javascript visualization libraries" [Computer software manual]. (R package version 0.4.5).

Wickham, Hadley. 2009. *ggplot2: Elegant graphics for data analysis.* Dordrecht: Springer.

Wilkinson, Leland. 2005. *The grammar of graphics.* 2. Aufl. New York: Springer.