Transkript zum Video DS1.1 Rohdaten und informationelle Komplexität

aus der Vorlesung Statistik für Wirtschaftswissenschaften

Inhalt

Hinweise

Zur Foliennummerierung: Das Transkript zum Video bezieht sich auf die im einfachen Foliensatz fortlaufenden Foliennummern 5-15.

Zur Schreibweise: Im Folgenden werden (sofern vorhanden) hochgestellte Zahlen oder Buchstaben durch ^ (A^2 = A^2) und tiefgestellte Zahlen oder Buchstaben durch _ (a_J = a_J) markiert.

Folie 1 – Deskriptive Statistik 1

Folientext

Deskriptive Statistik 1: Daten visualisieren und zusammenfassen. Alexander Silbersdorff, Wirtschaftswissenschaftliche Fakultät und Campus-Institut Data Science der Georg-August-Universität Göttingen, Logo der Georg-August-Universität Göttingen.

Folie 2 – Thema DS1.1

Folientext

Themen

DS1.1 Rohdaten und informationelle Komplexität

Sprechtext

Herzlich Willkommen zu diesem Lehrvideo der Veranstaltung Statistik zum Unterkapitel 'Deskriptive Statistik 1.1: Rohdaten und informationelle Komplexität'. In diesem Unterkapitel werde ich über die Darstellung von Rohdaten in Form von Urlisten sprechen, um den konzeptionellen Boden für die Erstellung von Häufigkeitsverteilungen und Maßzahlen in den Folgekapiteln zu legen.

Folie 3 – Lernziele

Folientext

Im Rahmen dieses Unterkapitels lernen Sie über...

- die Konzeption und Notation von Urlisten
- die Betrachtung von Urlisten über Ausprägungsmöglichkeiten
- die Strukturierung von Ausprägungsmöglichkeiten
- die Notwendigkeiten und Möglichkeiten von Komplexitätsreduktion bei der Betrachtung von Rohdaten

Sprechtext

Im Rahmen dieses Unterkapitels werden Sie insbesondere lernen, wie Rohdaten als Urlisten aufgefasst werden können und wie Urlisten notationell dargestellt werden können. Des Weiteren werden Sie lernen, wie Urlisten auf die vorliegenden Ausprägungsmöglichkeiten reduziert werden können, wie die Ausprägungsmöglichkeiten strukturiert werden können und nicht zuletzt, wie die Betrachtung von Urlisten über Ausprägungsmöglichkeiten eine oftmals notwendige Komplexitätsreduktion darstellt.

Folie 4 – Rohdaten – Merkmalsausprägungen und Urlisten

Folientext

- Univariate Betrachtung der Merkmale

Sprechtext

Einsteigen möchte ich mit einer formell abstrakten Betrachtung grundlegender Konzepte für Häufigkeitsverteilungen. Bei dieser Betrachtung nehmen wir vorerst eine univariate Perspektive ein, das heißt, wir betrachten vorerst immer nur eine Variable einzeln. In dem folgenden Deskriptive-Statistik-Kapitel DS2 werden wir dann diese univariate Perspektive auf eine multivariate Betrachtungsweise erweitern. Wir haben hier also eine Variable, welche wir der Konvention nach mit einem Großbuchstaben notieren, üblicherweise groß X, was ich hier auch noch mal handschriftlich eben festgehalten habe.

Folie 5 – Rohdaten – Merkmalsausprägungen und Urlisten 2

Folientext

- Rohdaten der Merkmalsausprägungen des Merkmals X in Spaltenvektor.
- Urliste:

- $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$
- (zeitweilige Einblendung 1: Wichtiger Begriff: Urliste bzw. Rohdaten)
- (zeitweilige Einblendung 2: Wichtiges Konzept: Rohdaten oft nicht direkt verständlich)

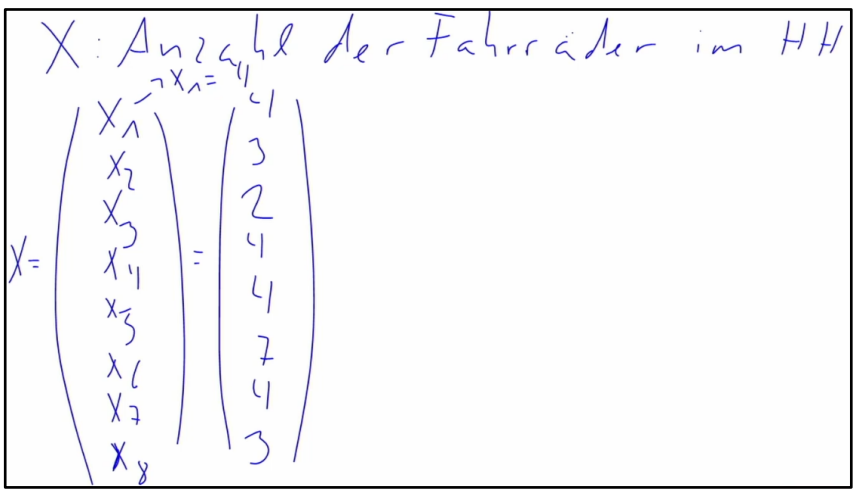
Sprechtext

Für dieses Merkmal X liegen uns eine Reihe von Merkmalsausprägungen vor: x_1, x_2 bis zu x_n, wobei wir hier nun klein x für die Merkmalsausprägung mit einem entsprechenden Index benutzen. Diese einzelnen Merkmalsausprägungen bzw. die Liste der Merkmalsausprägungen bezeichnen wir als Urliste oder einfach auch nur Rohdaten. Die Urliste können wir als Vektor verstehen, wobei wir üblicherweise im Rahmen dieser Veranstaltung einen Spaltenvektor verwenden. Sowohl die Begrifflichkeit der Urliste als auch die Notation in Form eines Spaltenvektors sollten Sie kennen. Und entsprechend ist die Begrifflichkeit Urliste bzw. Rohdaten hier nochmal separat eingeblendet und ich würde dann auch handschriftlich noch mal vermerken, dass es sich hier eben um einen Spaltenvektor handelt. Besonders hervorheben möchte ich, dass wir bei direkter Betrachtung der Rohdaten einen potenziell hochdimensionalen Vektorraum haben, den es kognitiv schlussendlich zu begreifen gilt. Das ist aber insbesondere bei großen Mengen an Rohdaten sehr komplex, bzw. dieser Vektorraum wird bei großen Mengen an Rohdaten hochdimensional, und das überschreitet im Regelfall schlichtweg unsere Hirnkapazitäten, was ich an der Stelle auch noch mal schriftlich vermerken würde. Dass diese Urlisten potenziell eben hochdimensional sind. Und aufgrund dieser Hochdimensionalität bzw. unserer eingeschränkten kognitiven Kapazitäten sind Rohdaten häufig bzw. im Regelfall nicht direkt verständlich. Lassen Sie mich dies anhand von einem Illustrationsbeispiel erläutern.

Folie 6 – Rohdaten – Merkmalsausprägungen und Urlisten 3

Folientext

- handschriftliche Notizen auf der Folie:



Sprechtext

Betrachten wir ein Merkmal, welches die Anzahl der Fahrräder in Göttinger Haushalten wiedergibt, also welches wir mit groß X notieren. Also groß X ist die Anzahl der Fahrräder im Haushalt. Lassen Sie uns jetzt vorerst annehmen, dass uns aus einer Datenerhebung Merkmalsausprägungen für acht der

insgesamt circa 80.000 Haushalte in Göttingen vorliegen. Das heißt, unsere Urliste ist ein Vektor mit acht Elementen. Diese Urliste kann zum einen abstrakt dargestellt werden, ohne dass ich die Werte kenne bzw. konkretisiere. Notationell könnte ich das Ganze jetzt wie folgt beschreiben: Ich habe x_1, x_2, x_3, x_4, x_5, x_6, x_7 und x_8, umrande das Ganze mit Vektorklammern und schreibe hier nochmal, dass ich den gesamten Vektor eben klein x nennen würde. Falls wir nun bereits konkrete Daten vorliegen haben, können wir anstelle der abstrakten Merkmalsausprägung natürlich auch die konkreten Werte betrachten. Das heißt, wenn wir annehmen, dass wir für den ersten Haushalt vier Fahrräder haben, bedeutet das, dass unser x_1 gleich 4 ist. Entsprechend wäre die erste Eintragung in unsere Urliste mit dem konkreten Wert die 4. Bei den weiteren Haushalten haben wir nun die Werte 3, 2, 4, 4, 7, 4 und 3, was ich hier jetzt auch noch einmal ausschreiben würde in den Vektor. Das heißt, das erste Element von unserem Vektor ist, wie gesagt, 4, das zweite Element in unserem Vektor ist die 3, das dritte Element ist die 2, das vierte Element wäre erneut eine 4, dann haben wir noch eine 4, unser sechstes Element wäre die 7, das siebte Element wäre die 4 und das letzte Element wäre die 3. Und diese Werte kann ich dann entsprechend auch in diesen Vektor schreiben und das wären dann entsprechend hier auch die konkreten Werte für unseren Beobachtungsvektor x. Und über diesen Vektor zeige ich jetzt, dass es sich bei den Daten um die Rohdaten, eben in Form eines Vektors, handelt. Natürlich sind Urlisten in der Praxis in den wenigsten Fällen derart überschaubar. Üblicherweise liegen deutlich mehr als acht Beobachtungen vor, wodurch die Urliste naturgemäß umfassender und damit auch unübersichtlicher wird.

Folie 7 – Komplexitätsreduktion – Illustrationsbeispiel

Folientext

- 128 Datenpunkte als Rohdaten
- Abbildung: Tabelle der Fahrradbestände von 128 Göttinger Haushalten

0	8	0	0	6	1	2	4	0	5	5	3
3	2	2	6	2	7	2	2	7	4	1	5
7	2	0	1	4	1	2	3	0	0	5	3
1	1	0	6	0	4	1	5	5	4	1	6
0	5	2	1	1	0	3	1	0	0	1	0
0	2	3	1	4	4	3	2	0	6	5	3
11	0	0	0	3	0	1	0	0	3	1	2
0	3	3	6	3	5	1	1	4	2	6	1
1	9	3	4	0	1	6	2	6	0	5	0
0	0	2	4	3	0	1	2	1	0	2	1
3	0	4	4	1	4	0	5				

Sprechtext

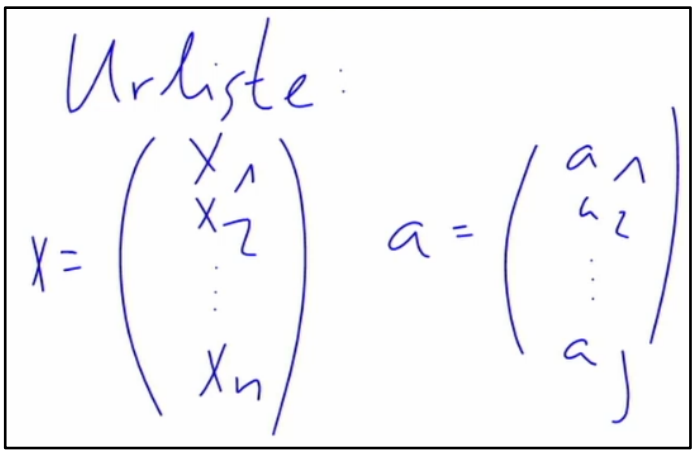
Für die Erhebung an Fahrradbeständen in Göttingen wurde in einer vor einigen Jahren durchgeführten Studie bspw. die Bestände von 128 Haushalten notiert. Die entsprechenden Daten mit den konkreten Werten sind jetzt hier auf dieser abgebildeten Folie dargestellt. Diese 128 abgebildeten Werte würden ein Vektor mit 128 Zeilen erfolgen. Das ist an dieser Stelle zum einen kaum vernünftig darzustellen und zum anderen noch viel weniger kognitiv einfach so zu begreifen. Um das Begreifen von diesen hier abgebildeten Daten soll es auch gar nicht gehen. Vielmehr soll dieses Beispiel hier verdeutlichen, dass bereits nach einem nach heutigen Maßstäben eigentlich sehr geringen, sehr moderaten Umfang von 128 Datenpunkten die Rohdaten kaum mehr direkt zu verstehen und prozessieren sind. Folglich ist es sinnvoll und in der Regel sogar notwendig, die Rohdaten so zu strukturieren und in einer Form zu präsentieren, dass sie für uns Menschen

verständlich und prozessierbar sind. Eine Möglichkeit dies zu tun, ist, eine Häufigkeitsverteilung zu erstellen.

Folie 8 – Betrachtung über Ausprägungsmöglichkeiten – Notation und Skalenniveaus

Folientext

- a_1, ..., a_J als die Ausprägungsmöglichkeiten des Merkmals X.
- handschriftliche Notizen auf der Folie:



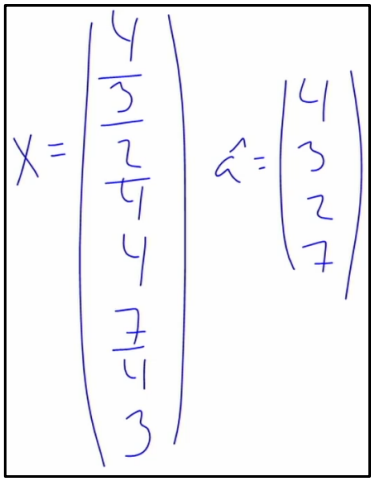
Sprechtext

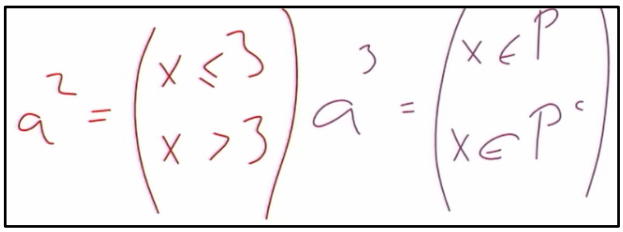
Die Form der Häufigkeitsverteilung, zu welcher wir im folgenden Unterkapitel im Konkreten noch mal kommen werden, wird formell durch eine Menge strukturiert, welche wir auch in Form eines Vektors repräsentieren können. Diese strukturierende Menge sind die Ausprägungsmöglichkeiten des Merkmals X. Und die Ausprägungsmöglichkeiten notieren wir im Rahmen dieser Veranstaltung allgemein mit klein a bzw. für die einzelnen Elemente mit einem Index, welcher von 1 bis groß J läuft. Das heißt, neben der Menge der Urliste habe ich eine zweite Menge, die Menge der Ausprägungsmöglichkeiten. Für den abstrakten Fall habe ich also auf der einen Seite die Urliste mit meinen Beobachtungen x_1 bis x_n. Oder hier eben klein x für die gesamte Urliste, und dazu das abstrakte Analogon der Ausprägungsmöglichkeiten, welches dann hier klein a ist und welches von a_1 über a_2 hin zu a_J läuft. Vorerst nehmen wir jetzt der Einfachheit halber an, dass wir als Menge der Ausprägungsmöglichkeiten alle in der Urliste enthaltenen Beobachtungen berücksichtigen. Grundsätzlich kann jedoch dieser Ausprägungsmöglichkeitsvektor jedwede Form annehmen. Das heißt, in dem Fall von den numerischen Variablen, wie in unserem Illustrationsbeispiel, nehmen wir jede Zahl, die mindestens einmal in unserer Urliste vorkommt, als Ausprägungsmöglichkeit auf.

Folie 9 – Betrachtung über Ausprägungsmöglichkeiten – Notation und Skalenniveaus 2

Folientext

- handschriftliche Notizen auf der Folie:





Sprechtext

Betrachten wir jetzt unser Illustrationsbeispiel mit der Urliste der acht Merkmalsausprägungen, so haben wir einen Vektor mit acht Elementen, welchen ich hier nochmal erneut aufschreiben würde. Also ich hab unser klein x, den Vektor unserer Ausprägungsmöglichkeiten, und habe dann hier als konkrete Zahlen die Zahlen 4, 3, 2, 4, 4, 7, 4, 3 und ziehe hier die Vektorklammern durch. Und für diese Rohdaten können wir nun einen Vektor der Ausprägungsmöglichkeiten definieren. Nach der eben genannten default-Regel enthält dieser Vektor die Menge aller in den Rohdaten enthaltenen Ausprägungsmöglichkeiten. Betrachten wir die hier dargestellte Urliste, so haben wir also als Ausprägungsmöglichkeiten in unserem Vektor, den ich jetzt hier mit klein a notiere, zum einen die Zahl 4, dann ist hier die 3 aufgeführt, die 2 aufgeführt, die 4 haben wir bereits aufgeführt, das heißt wir müssen sie nicht nochmal hinzunehmen, haben dann als nächste neue aufgeführte Zahl die 7. Danach werden erneut nur Zahlen aufgeführt, die bereits hier in unserer Liste und unseren Vektor der Ausprägungsmöglichkeiten aufgenommen wurden. Bevor ich jetzt mit diesem Ausprägungsmöglichkeitsvektor weitermache, möchte ich noch zwei alternative Darstellungen bzw. zwei alternative Möglichkeiten für den Vektor der Ausprägungsmöglichkeiten aufzeigen, denn grundsätzlich können die Ausprägungsmöglichkeiten aus letztendlich jeder sich eindeutig abgrenzenden Kategorisierung bestehen. Bspw. können wir hier jetzt als Alternative, welche ich farblich nochmal in rot schreiben würde, oder wir haben hier unseren Ursprungsausprägungsmöglichkeitsvektor, der war in blau, würde ich hier nochmal mit einer 1 notieren. Dann würde ich jetzt einen zweiten alternativen Ausprägungsmöglichkeitsvektor nochmal definieren, welchen ich mit a^2 notieren würde, welcher bspw. jetzt zwei Kategorien enthält. Nämlich einmal die Kategorie, dass unsere jeweilige Beobachtung, die wir vorliegen haben, dass unser x, ein gegebenes Element aus unserem x-Vektor, dass das kleiner-gleich 3 ist. Also dass ich bis zu drei Fahrräder in meinem Haushalt habe, oder dass ich als Alternative dazu bspw. mehr als drei Fahrräder habe. Und das wäre eine alternative Darstellung oder ein alternativer Ansatz für die Darstellung der Ausprägungsmöglichkeiten. Und natürlich gibt es noch weitere Optionen. Ich könnte bspw. hier als dritte Option, die ich aufführen würde und jetzt in der dritten Farbe darstelle, bzw. den Vektor der Ausprägungsmöglichkeiten hier mit a^3 notiere, wobei ich das 3 eben, wie Sie hier sehen, jetzt nicht im Index schreibe, um da keine Verwirrung aufkommen zu lassen, sondern eben oben.

Dort könnte ich jetzt bspw. meine Ausprägungsmöglichkeiten in Form einer anderen Menge strukturieren, oder zwei Mengen, einer Menge von Mengen, wo ich hier an dieser Stelle bspw. sagen würde: Okay, ich habe jetzt hier zum einen die Menge der Primzahlen als eine Ausprägungsmöglichkeit. Also ich sage letzten Endes, dass mein x ein Element der Primzahlen ist, der Menge der Primzahlen, und als alternative Menge dazu könnte ich dann sagen: Okay, es ist Element von der Komplementärmenge zu den Primzahlen, das heißt der Menge all jener Zahlen, die nicht Primzahlen sind. Und auch das könnte ich hier verwenden und P würde hier, wie gesagt, für die Menge der Primzahlen stehen. Was an dieser Stelle natürlich etwas hochgestochener mathematischer ist und was letzten Endes aufzeigen soll, dass ich natürlich über formelle Notationen eine Vielschichtigkeit an Mengen definieren kann, die, wie wir noch besprechen werden im weiteren Verlauf, wo wir sehen werden, dass diese Primzahlen eben im Gegensatz zu der davor definierten Menge a^2 eben keine konvexe Menge ist, was an manchen Stellen eben recht hilfreich ist. Über diese zwei anderweitigen Formulierungen werden wir im Verlauf gleich, wie gesagt, noch mal sprechen. Jetzt vorerst aber kehren wir zurück zu ebendieser ersten Ausprägungsmöglichkeit, die wir üblicherweise als eine Art Default betrachten.

Folie 10 – Betrachtung über Ausprägungsmöglichkeiten – Notation und Skalenniveaus 3

Folientext

- a_1, ..., a_J als die Ausprägungsmöglichkeiten des Merkmals X.
- Für mindestens ordinale Merkmale seien diese Werte darüber hinaus geordnet, also a_1 < ... < a_J.
- handschriftliche Notizen auf der Folie:
  - ungeordnet (4, 3, 2, 7)
  - geordnet (2, 3, 4, 7)
- Wechsel auf die vorherige Folie (siehe Folie 9). Unter den ersten beiden Vektoren wird handschriftlich ergänzt: n = 8 (unter dem Vektor der Rohdaten x), J = 4 und J < n (unter dem Vektor der Ausprägungsmöglichkeiten a^1).

Sprechtext

Grundsätzlich gilt das, wenn die Ausprägungsmöglichkeiten eine ordinale Struktur aufweisen, das heißt, wenn sie eine Struktur haben, die in eine sinnvoll zu interpretierende aufsteigende Reihenfolge gebracht werden kann. Dann ist es üblich, ist es die Konvention, die Ausprägungsmöglichkeiten entsprechend aufsteigend zu ordnen. Das heißt, für unser Illustrationsbeispiel würden wir die Ausprägungsmöglichkeiten, die wir in der Ursprungsform hatten, die in ungeordneter Form eben 4, 3, 2, 7 waren, könnten und sollten wir der Konvention nach eben ordnen. Das heißt, geordnet würden wir jetzt diese Zahlen, die wir hier in unserem Ausprägungsmöglichkeitsvektor haben, üblicherweise in aufsteigender Reihenfolge schreiben, das heißt 2, 3, 4, 7, um das Ganze besser zu strukturieren und übersichtlicher zu machen oder unseren Gedankengängen anzupassen, die üblicherweise aufsteigend denken. Wenn wir jetzt keine solche ordinale Struktur haben, das heißt, wenn Sie statt einer ordinalen Variable oder einer mindestens ordinal strukturierten Variable eine nominale Variable betrachten, gibt es in der Regel keine sich natürlich ergebende Ordnungsstruktur, sodass Sie letztendlich mit jeder Ordnungsstruktur arbeiten können, welche Ihnen zweckdienlich erscheint. Das heißt, ob Sie bspw. bei dem Primzahlenbeispiel mit den Primzahlen einsteigen oder sagen, dass das Komplement zu den Primzahlen den Einstieg darstellt, dort gibt es keine sich natürlich ergebende Struktur und dementsprechend können Sie hier die Struktur letzten Endes frei nach ihrem Gusto wählen. Bei mindestens ordinalskalierten Variablen,

das heißt auch kardinalskalierten Variablen, dort sollten Sie aber die Ordnungsstruktur im Zweifelsfall betrachten.

(Wechsel auf die vorherige Folie (siehe Folie 9))

Abschließend möchte ich an dieser Stelle noch einmal explizit darauf hinweisen, dass die Anzahl der Elemente von dem Rohdatenvektor in unserem Fallbeispiel größer ist als die Anzahl der Elemente in unserem Vektor der Ausprägungsmöglichkeiten. Oder anders ausgedrückt, der Vektor der Ausprägungsmöglichkeiten ist weniger umfangreich als der Vektor der Urliste. Wenn wir die Anzahl der Elemente in unseren Rohdaten also mit n notieren, das heißt hier n, was bei unseren Rohdaten acht Beobachtungen wären, also n=8, und die Menge der Elemente in unserem Vektor der Ausprägungsmöglichkeiten mit J schreiben, was in unserem Fall vier Ausprägungsmöglichkeiten sind, so gilt in unserer Relation oder in unserem Illustrationsbeispiel, das J strikt kleiner ist als n.

Folie 11 – Betrachtung über Ausprägungsmöglichkeiten – Illustrationsbeispiel

Folientext

- Abbildung: Tabelle der Fahrradbestände von 128 Göttinger Haushalten (siehe Folie 7)
- $$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \\ a_{10} \\ a_{11} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 11 \end{pmatrix}$$
- handschriftliche Notizen auf der Folie:
  - J = 11 (über dem Vektor)
  - n = 128 (über der Tabelle)
  - J << n

Sprechtext

Und diese Ungleichheit ist natürlich noch eklatanter für das volle Fallbeispiel mit den 128 Beobachtungen, wo der Vektor der Ausprägungsmöglichkeiten, den Sie hier abgebildet sehen mit a, elf Elemente umfasst. Das heißt, J ist hier gleich 11. Das schreibe ich über den Vektor. Und auf der anderen Seite haben wir bei den Rohdaten eben eine Anzahl von 128 Elementen, das heißt, hier haben wir n=128, was ich über die Tabelle schreibe. Und somit ist die Anzahl der Elemente in unserem Vektor der Ausprägungsmöglichkeiten im Vergleich zum Vektor mit den Rohdaten um 90 Prozent reduziert. In diesem Fall können wir folglich nicht nur sagen, dass J strikt kleiner ist als n, sondern darüber hinaus gehend kann man mit gutem Gewissen argumentieren, dass J sogar viel kleiner ist als n, was ich hier mit zwei hintereinander geschriebenen Kleiner-als-Zeichen, also J<<n, notiere. Und dieser Zusammenhang - J ist kleiner als n bzw. J ist viel kleiner als n - gilt in den meisten Fällen, bzw. ist letzten Endes ein fundamentales Ziel der Betrachtung von Rohdaten mittels Häufigkeitsverteilungen. Denn Vektoren mit weniger Ausprägungsmöglichkeiten sind kognitiv einfacher zu greifen, können wir Menschen besser verstehen. Das heißt, was wir an dieser Stelle

erreicht haben bzw. worauf wir abgezielt haben, ist ein erster Schritt hin zu einer informationellen Reduktion. Indem wir Rohdaten auf der Grundlage der wie auch immer definierten Merkmalsausprägungen heraus betrachten, reduzieren wir die Komplexität des betrachteten Vektorraums von n auf J. Denn im hier dargestellten Illustrationsbeispiel reduzieren wir die Dimensionalität von 128 auf 11. Der Preis für die durch diese Reduktion gewonnene Übersichtlichkeit ist jedoch, dass Informationen aus den Rohdaten verloren gehen, wie bspw. hier die Reihenfolgestruktur der Merkmalsausprägungen der Urliste, die ggf. von Interesse ist. Die Betrachtung von Rohdaten über Ausprägungsmöglichkeiten bzw. dann anschließend über Häufigkeitsverteilungen, wozu wir im nächsten Unterkapitel kommen werden, stellt somit eine Betrachtungsform der Rohdaten dar, welche auf einer informationellen Reduktion beruht. Dies bringt auf der einen Seite Vorteile mit sich, bringt aber auf der anderen Seite, wie jede Reduktion, auch potenzielle Probleme mit sich, welche wir an späterer Stelle nochmal diskutieren.

Folie 12 – Betrachtung über Ausprägungsmöglichkeiten – Umsetzung in R

Folientext

Folgende R-Befehle werden im Rahmen dieses Unterkapitels eingeführt:

1. unique()
2. cut()
3. setdiff()
4. ifelse()

Sprechtext

Bevor wir dieses Unterkapitel jetzt abschließen, würde ich im Rahmen dieses Unterkapitels Ihnen im Folgenden jetzt nochmal drei R-Befehle vorstellen bzw. auf drei R-Befehle eingehen wollen, welche Sie nutzen können, um Ausprägungsmöglichkeiten in Urlisten zu betrachten.

1. Urliste in R erstellen

Text in R

Scriptfenster	# Urliste
	x <- c(4,3,2,4,4,7,4,3)

Ausgabe Konsole	> x <- c(4,3,2,4,4,7,4,3)

Sprechtext

Wenngleich die Betrachtung über R eigentlich erst bei umfangreicheren Urlisten notwendig wird, würde ich das Ganze hier jetzt anhand des Beispiels mit nur acht Beobachtungen illustrieren, um es überschaubar zu halten. Was ich jetzt zuerst mache, ist, dass ich die Urliste eingebe. Entsprechend tippe ich erstmal ins Scriptfenster "# Urliste", damit das Ganze vernünftig strukturiert ist, und dann gebe ich die Urliste ein, indem ich sage "x", mein Vektor mit der Urliste oder bzw. Vektorobjekt, in dem ich die Urliste hinterlegen möchte, den Zuweisungspfeil ("<-") und dann eben klein "c", der combine-Befehl, um dann eben die einzelnen Elemente, die Zahlen, die wir haben, in einem Spaltenvektor zusammenzufassen. Und in die Klammern hinter c gebe ich entsprechend die Merkmalsausprägungen ein, das heißt 4, 3, 2, 4, 4, 7, 4 und 3. Und dann muss die Klammer zugesetzt werden, bzw. in RStudio ist das Ganze automatisch, und setze das Ganze dann über Strg+Enter um, und Sie müssen natürlich darauf achten, dass ihre Kommas richtig gesetzt sind. Entsprechend habe ich nun die Rohdaten über den Befehl c() als Vektorobjekt, genauer gesagt ein Spaltenvektor, unter dem Objektnamen klein x hinterlegt.

2. Kategorisierungsvarianten an Ausprägungsmöglichkeiten

Text in R

Scriptfenster	# Ausprägungsmöglichkeiten mit R: v1 # Ausprägungsmöglichkeiten mit R: v2 # Ausprägungsmöglichkeiten mit R: v3

Sprechtext

Und für diese unter x hinterlegte Urliste möchte ich eben nun drei verschiedene Kategorisierungsvarianten an Ausprägungsmöglichkeiten definieren bzw. Ihnen drei Varianten vorstellen. Was ich also mache, ist, dass ich schreibe, ich habe hier "Ausprägungsmöglichkeiten mit R" hinter einem "#" und hier habe ich quasi "v1" für Variante 1, und würde das Ganze hier noch mal mit "v2" hinschreiben als Überschrift und mit "v3", damit das Ganze im Nachhinein vernünftig strukturiert ist.

3. Kategorisierungsvariante 1: Der unique()-Befehl

Text in R

Scriptfenster	# Ausprägungsmöglichkeiten mit R: v1 - unique()
	unique(x)
	(a1 <- unique(x))

Ausgabe Konsole	> # Ausprägungsmöglichkeiten mit R: v1 - unique() > unique(x) [1] 4 3 2 7 > # Ausprägungsmöglichkeiten mit R: v1 - unique() > (a1 <- unique(x)) [1] 4 3 2 7
Zeitweilige Einblendung	Wichtiger R-Befehl: unique()

Sprechtext

Die erste Variante, welche mehr oder weniger als Default zu betrachten ist und worauf wir in unseren Illustrationsbeispielen den Fokus gelegt haben, ist, alle Ausprägungsmöglichkeiten ungruppiert zu betrachten. Um sich im Rahmen einer solchen Betrachtungsweise die resultierenden Ausprägungsmöglichkeiten automatisch ausgeben zu lassen, können Sie den Befehl unique() aus dem Base-Paket benutzen. Das heißt, die Variante 1 fokussiert sich auf den unique()-Befehl, was ich hier entsprechend dann auch nochmal in der Überschrift mit "unique()" vermerken würde. Und dieser unique()-Befehl ist nützlich und ist auch wichtig, und daher wird der jetzt hier separat auch nochmal eingeblendet. Um den unique()-Befehl zu nutzen, tippe ich schlussendlich einfach nur "unique" ein, Klammer auf und füge dort das Objekt "x" ein, auf den dieser unique()-Befehl angewendet werden soll. Was der unique()-Befehl dann macht, ist, dass er letztendlich alle duplizierten Elemente im Vektor entfernt, sodass schlussendlich nur ein Vektor übrig bleibt, in dem alle Ausprägungsmöglichkeiten jeweils einmal vorkommen. Das heißt, ich könnte diesen Vektor ausführen bzw. alternativ dazu könnte ich das Ganze meinetwegen im Objekt a1 hinterlegen und könnte hier bspw. auch noch die Klammern direkt drum setzen um diesen Befehl, damit mir das Ganze auch noch direkt ausgegeben wird. Sonst ist das Ganze nur in dem Objekt oder der resultierende Vektor der Ausprägungsmöglichkeiten ist in dem Objekt a1 hinterlegt. Bei dem Output ist jetzt zu beachten, dass die Ordnung aus dem ausgegebenen Vektor der Reihenfolge der Nennung in der Urliste entspricht. Das heißt, wir haben als Ausgabe eben 4, 3, 2 und 7, wie das auch im analogen Vorgehen zuerst der Fall war.

3.1. Einschub: Der sort()-Befehl

Text in R

Scriptfenster	(a1.s <- sort(a1))
Ausgabe Konsole	> (a1.s <- sort(a1)) [1] 2 3 4 7

Sprechtext

Falls nötig und falls möglich, kann der über unique() erzeugte Vektor, sofern die entsprechende Ordnungsstruktur in R hinterlegt ist, das heißt, sofern das Ganze ordinalskaliert ist und auch quasi mit einer Ordnungsstruktur in R hinterlegt ist, was bei numerischen Objekten natürlich automatisch der Fall ist, über den Befehl sort() in eine aufsteigende Reinfolge gebracht werden. Um das umzusetzen, würde ich jetzt hier bspw. sagen "a1.s" für den sortierten Vektor a1, und würde dann eben den Befehl "sort()" auf diesen Vektor "a1" anwenden, das Ganze mit dem Zuweisungspfeil machen, und ich würde hier direkt noch mal die Klammern setzen, damit Sie gleich den Output sehen, und das Ganze ausführen. Und dann sehen wir hier, dass eben über diesen sort()-Befehl die aufsteigende Reihenfolge generiert wurde, also 2, 3, 4 und dann 7.

3.2. Verknüpfung von unique()- und sort()-Befehl

Text in R

Scriptfenster	a1 <- sort(unique(x)) a1
Ausgabe Konsole	> a1 <- sort(unique(x)) > a1 [1] 2 3 4 7

Sprechtext

Wenn Sie das Ganze nicht in zwei separaten Schritten machen wollen, mit einer Objektzwischenspeicherung, können Sie die beiden Befehle natürlich auch in eins miteinander verschachteln und dann umsetzen. Das heißt, ich könnte alternativ hierzu direkt "a1" tippen, also ich würde dieses Objekt a1 einfach überschreiben, was ich ursprünglich erzeugt habe, was noch ungeordnet war. Ich würde schreiben "sort()" auf das Objekt, was über unique() erzeugt wird, wenn ich "unique(x)" eingebe, und dann wird das Ganze eben hintereinander verschachtelt, direkt angewandt in einer Zeile. Und der resultierende Vektor a1, also die Klammern mache ich jetzt mal nicht, damit es einfach hinterlegt ist, und wenn ich mir dann den Vektor a1 angucke, der ursprünglich ungeordnet war, sehen Sie, dass jetzt dieser Vektor hier geordnet ist, was im Zweifelsfall zu empfehlen ist.

4. Kategorisierungsvariante 2: Der cut()-Befehl

Text in R

Scriptfenster	# Ausprägungsmöglichkeiten mit R: v2 - cut()
Zeitweilige Einblendung	Wichtiger R-Befehl: cut()

Sprechtext

Kommen wir nun zur zweiten Variante. Die zweite Variante basiert oder fokussiert auf den cut()-Befehl, welcher auch im Base-Paket zu finden ist, und entsprechend würde ich hier auch nochmal den Befehl "cut()" in der Überschrift vermerken. Und auch dieser Befehl ist entsprechend wichtig, ist jetzt auch hier nochmal eingeblendet. Den sollten Sie entsprechend können und damit umgehen

können. Was dieser cut()-Befehl macht, ist, dass er die Elemente aus einem numerischen Vektor vorab definierten Intervallen zuordnet und diese Intervalle bzw. die Ausprägungsmöglichkeiten oder die Urliste, die dann entsprechend diesen Ausprägungsmöglichkeiten umstrukturiert wird, als Faktorobjekt abspeichert. Somit erlaubt jetzt dieser cut()-Befehl die Gruppierung von Elementen in einem numerischen Vektor, was üblicherweise dann sinnvoll bzw. notwendig wird, wenn die Rohdaten zu viele unterschiedliche Ausprägungsmöglichkeiten aufweisen, um sie, wie in der eben gezeigten Variante 1, alle einzeln zu betrachten. Das ist insbesondere bspw. bei kardinalskalierten Variablen der Fall, wo Sie möglicherweise Nachkommastellen haben usw.

4.1. Grundstruktur/Syntax vom cut()-Befehl

Text in R

Scriptfenster	(a2 <- cut(x, c(0,2,Inf),include.lowest=TRUE)) (a2 <- cut(x,c(0,2,Inf),right=FALSE)) (a2 <- cut(x,c(0,exp(1),pi,7),right=FALSE,include.lowest=TRUE)) (a2 <- cut(x,c(0,exp(1),pi,7),right=FALSE,include.lowest=TRUE, labels=c("<Euler","Euler-Pi",">Pi")))
Ausgabe Konsole	> (a2 <- cut(x, c(0,2,Inf),include.lowest=TRUE)) [1] (2,Inf] (2,Inf] [0,2] (2,Inf] (2,Inf] (2,Inf] (2,Inf] (2,Inf] Levels: [0,2] (2,Inf] > (a2 <- cut(x,c(0,2,Inf),right=FALSE)) [1] [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf) Levels: [0,2) [2,Inf) > (a2 <- cut(x,c(0,exp(1),pi,7),right=FALSE,include.lowest=TRUE)) [1] [3.14,7] [2.72,3.14) [0,2.72) [3.14,7] [3.14,7] [3.14,7] [3.14,7] [2.72,3.14) Levels: [0,2.72) [2.72,3.14) [3.14,7] > (a2 <- cut(x,c(0,exp(1),pi,7),right=FALSE,include.lowest=TRUE, labels=c("<Euler","Euler-Pi",">Pi"))) [1] >Pi Euler-Pi <Euler >Pi >Pi >Pi >Pi Euler-Pi Levels: <Euler Euler-Pi >Pi

Sprechtext

Um Ihnen jetzt einige Feinheiten des cut()-Befehls aufzuzeigen, will ich im Folgenden vier Variationen betrachten, welche ich der Einfachheit halber jetzt einfach aus einem vorbereiteten Skript herauskopiere und in mein Arbeitsskript hier wieder reinkopiere und dann gleich ausführe. Bevor ich das mache, würde ich noch einmal auf die Gemeinsamkeiten bei diesen Befehlen eingehen bzw. die Grundstruktur von diesem cut()-Befehl erläutern. Das erste Argument im cut()-Befehl sollte ein numerischer Vektor sein, in unserem Fall jetzt immer der numerische Vektor x mit den acht Beobachtungen, die wir eben hier eingetragen haben. Das heißt, hier sehen wir, das erste Argument ist das x. Das zweite Argument in diesem cut()-Befehl sind dann die Bruchpunkte, im Englischen 'breaks', zwischen den einzelnen Intervallen bzw. die Bruchpunkte, welche die einzelnen Intervalle beschreiben. Das erste und das letzte Argument sind die Randpunkte der äußeren Intervalle, während alle inneren Elemente dann die inneren Intervallgrenzen, also die Bruchpunkte zwischen den Intervallen, definieren. Und auf diese Bruchpunkte werde ich gleich im Detail noch mal bei den einzelnen Beispielen eingehen. Und zuletzt haben wir dann hier verschiedene Optionen, die Sie hier sehen, die eingeführt werden, und auf diese weitergehenden Optionen würde ich dann auch nochmal im Einzelfall eingehen. Entsprechend würde ich diese vier Befehle, welche wir jetzt hier sehen, einmal ausführen über Strg+Enter.

4.2. cut()-Befehl Syntax Variante 1

Text in R

Ausgabe Konsole	> (a2 <- cut(x, c(0,2,Inf),include.lowest=TRUE)) [1] (2,Inf] (2,Inf] [0,2] (2,Inf] (2,Inf] (2,Inf] (2,Inf] (2,Inf] Levels: [0,2] (2,Inf]

Sprechtext

Beim ersten Befehl, welcher hier ganz oben jetzt ausgegeben wird in der Konsole, setzen wir nach dem numerischen Vektor x im Befehl einen Vektor, der die breaks definiert, welcher uns drei breaks gibt und somit zwei Intervalle definiert: ein Intervall von 0 bis 2 und ein weiteres Intervall, was dann von 2 bis unendlich geht. Und dieses 'unendlich' ist konzeptionell oder auf eine gewisse Art und Weise unter der speziellen Begrifflichkeit 'Inf' hinterlegt. Und als letztes, als dritte Option, setzen wir jetzt hier die Option include.lowest auf TRUE. Und durch dieses auf TRUE Setzen definieren wir die unterste Intervallgrenze als geschlossenes Intervall. Das heißt, im resultierenden Output werden für diesen ersten Befehl zwei Levels generiert, zwei Ausprägungsmöglichkeiten: Die erste Ausprägungsmöglichkeit von 0 bis 2 mit geschlossenen Klammern auf beiden Seiten, das heißt, wir haben hier eine geschlossene Menge, in der beide Endpunkte, die zwei Randpunkte für dieses Intervall inkludiert sind, 0 und 2. Das zweite Intervall dagegen ist offen bzw. halboffen. Die untere Intervallgrenze wird hier mittels runder Klammer als offen gekennzeichnet und durch diese offene Intervallgrenze wird sichergestellt, dass die zwei Intervalle disjunkt sind. Das heißt, die zwei selbst, dieser Breakpoint, der Bruchpunkt, wird nur dem ersten Intervall und nicht dem zweiten Intervall zugeordnet. Und auf der anderen Seite des Intervalls wird bei Inf an dieser Stelle eine geschlossene Klammer gesetzt. Auf die Details, warum wir hier bei Inf eine geschlossene Klammer haben und warum Sie nicht, wie Sie das in Mathe hoffentlich gelernt haben, bei Unendlichkeiten eigentlich immer offene Klammern zu nutzen sind, möchte ich jetzt nicht allzu ausführlich eingehen. Für diejenigen, die es tiefer interessiert, sei unter anderem darauf hingewiesen, dass letztendlich Inf in R eben nicht mit der theoretischen Unendlichkeit oder dem theoretischen Unendlichkeitskonzept aus der Mathe gleichzusetzen ist, sondern dass letztendlich konzeptionell jene Menge an großen Zahlen darin subsumiert wird, die in R eben nicht durch das genutzte Double-precision floating-point format dargestellt werden kann. Wenn Sie das interessiert, wenn Sie dem nachgehen wollen, was mich freuen würde, können Sie hier in einschlägiger Literatur zu R, zum Konzept von Inf und dann dem cut()-Befehl mehr recherchieren. Es würde sich aber eben nur an diejenigen richten, die explizite Topnoten anstreben. Der Rest kann das getrost vernachlässigen. Wichtig für alle ist hingegen, dass in R der gesamte Vektor, das heißt unsere Urliste bzw. die Ausgabe der Urliste, jetzt über diesen cut()-Befehl bereits entsprechend der zwei definierten Kategorien bzw. der definierten Ausprägungsmöglichkeiten umdefiniert wurde. Das heißt, was passiert, wenn wir cut() nutzen, ist, dass eine Informationsreduktion stattfindet, da die Urliste nun entlang der im cut()-Befehl definierten Ausprägungsmöglichkeiten dargestellt wird. Und in unserem Fall bzw. grundsätzlich gilt, dass in diesem Schritt Komplexität reduziert wird, da alle Zahlen, alle Elemente der Urliste jetzt umdefiniert werden in auf der einen Seite die Kategorie zwischen 0 und 2, oder dass alle Zahlen in der Urliste entweder in diese Kategorie 0 bis 2 subsumiert werden oder aber auf der anderen Seite in jene Zahlen, die größer sind als 2, subsumiert werden. Und das ist letztendlich eine informationelle Reduktion.

4.3. cut()-Befehl Syntax Variante 2

Text in R

Ausgabe Konsole	> (a2 <- cut(x,c(0,2,Inf),right=FALSE)) [1] [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf) [2,Inf)

	Levels: [0,2) [2,Inf)

Sprechtext

Beim Output des zweiten Befehls ist zu sehen bzw. festzuhalten von meiner Seite, dass über diesen Befehl right=FALSE die Ordnungsstruktur der Intervalle geändert wurde. Das heißt, dass jetzt das erste Intervall, was davor nach oben hin geschlossen war, jetzt an dieser Stelle hier nach oben hin geöffnet ist, sodass die Intervallgrenzen anders gesetzt sind und der Breakpoint, die 2 selber, jetzt dem oberen Intervall zugeschlagen wird. Entsprechend können Sie durch die Option right im cut()-Befehl die Öffnungsstruktur der Intervalle, welche Sie durch Breaks setzen, bestimmen. Wenn Sie right=TRUE setzen, was der Default ist, sind die rechten Intervallgrenzen bzw. die oberen Intervallgrenzen im Regelfall geschlossen und die unteren entsprechend offen. Wenn Sie right=FALSE setzen, wie wir das im zweiten Beispiel gemacht haben, sind die oberen Intervallgrenzen im Regelfall offen und entsprechend die unteren geschlossen.

4.4. cut()-Befehl Syntax Variante 3 und 4

Text in R

Ausgabe Konsole	> (a2 <- cut(x,c(0,exp(1),pi,7),right=FALSE,include.lowest=TRUE)) [1] [3.14,7] [2.72,3.14) [0,2.72) [3.14,7] [3.14,7] [3.14,7] [3.14,7] [2.72,3.14) Levels: [0,2.72) [2.72,3.14) [3.14,7] > (a2 <- cut(x,c(0,exp(1),pi,7),right=FALSE,include.lowest=TRUE, labels=c("<Euler","Euler-Pi",">Pi"))) [1] >Pi Euler-Pi <Euler >Pi >Pi >Pi >Pi Euler-Pi Levels: <Euler Euler-Pi >Pi

Sprechtext

Dann gibt es noch den dritten und vierten Befehl, den Sie hier sehen. Bei diesem dritten und vierten Befehl nutzen wir nun andere Intervallgrenzen. Intervallgrenzen, die auch irrationale Zahlen beinhalten, wie an dieser Stelle Pi, was in R mit "pi" hinterlegt ist, und die Eulersche Zahl, also die Zahl e, was e^1 entspricht, was wir über den Befehl exp() realisieren können. Damit führen wir eben letzten Endes nochmal neue Intervallgrenzen ein und würden hier in diesem zweiten und dritten Befehl auch nochmal neue Kombinationen darstellen für die Option right und include.lowest. Der Unterschied zwischen dem dritten und dem vierten Befehl, den Sie hier aufgeführt sehen, ist dann, dass wir über diesen cut()-Befehl, wie gesagt, ein factor-Objekt definieren und dass wir in dem vierten Befehl mittels der labels-Option, welche im factor()-Befehl inkludiert ist, die Möglichkeit der Kennzeichnung der Intervalle, der Labels, welche dann mittels character string beschrieben werden können, wahrnehmen, um zu sagen, das erste Intervall sind eben all jene Zahlen, die kleiner sind als die Eulersche Zahl. Dann haben wir ein Intervall, was die Zahlen zwischen der Eulerschen Zahl und Pi berücksichtigt, und dann haben wir alle Zahlen, die größer sind als Pi. Was ich mit diesen zwei Optionen darstellen wollte, sind noch mal weitergehende Variationen, die wir nutzen können in dem cut()-Befehl. Diese fortgeschrittenere Nutzung bzw. diese mathematischen, hochgestochenen Intervallgrenzen usw. würde ich aber denjenigen überlassen, die gute oder sehr gute Noten anstreben, und es von meiner Seite eben an der Stelle auch dabei belassen, über diese zwei weiteren Varianten zu reden.

5. Kategorisierungsvariante 3: Der ifelse()-Befehl

Text in R

Scriptfenster	# Ausprägungsmöglichkeiten mit R: v3 - ifelse()

Zeitweilige Einblendung	Wichtiger R-Befehl: ifelse()

Sprechtext

Ich möchte nun abschließend zur dritten Ausprägungsmöglichkeit, zum dritten Befehl, übergehen, welcher ein Befehl ist, der es uns erlaubt, Ausprägungsmöglichkeiten noch allgemeiner zu definieren. Dabei würde ich mich hier im Rahmen dieser Veranstaltung darauf beschränken, dass wir einfach zwei generell definierte Mengen als Ausprägungsmöglichkeiten haben, die disjunkt sind, welche an dieser Stelle so definiert werden, dass sie eine bestimmte Zuordnungsbedingung erfüllen oder aber eben nicht erfüllen. Und diese Zuordnungsbedingung kann mehr oder weniger beliebig gestaltet sein und kann bspw., wie in dem jetzt folgenden Fallbeispiel, unter anderem auch nichtkonvexe Mengen beinhalten, was im Gegensatz zu dem cut()-Befehl, wo das Ganze nicht nötig ist, hier eben über diesen Befehl möglich wäre und auch noch mal die Flexibilität dieses Befehls andeutet. Der hierfür benutzte Befehl, welcher, wie gesagt, ein ziemlich flexibler Befehl ist, der auch in ganz vielen anderen Kontexten eingesetzt werden kann, ist der ifelse()-Befehl, was ich hier auch nochmal in der Überschrift vermerke. Diesen Befehl sollten Sie, wie die anderen Befehle auch, entsprechend kennen. Er ist explizit Teil dieses Moduls, ist entsprechend hier auch nochmal eingeblendet unten und wird vielschichtig verwendet.

5.1. Logische Abfrage und Aufbau/Syntax des ifelse()-Befehls

Sprechtext

Ich würde diesen Befehl jetzt nutzen, um eine logische Abfrage zu machen, die die Elemente unseres Vektors x eben zwei unterschiedlichen Mengen als Ausprägungsmöglichkeit zuordnet, zum einen der Menge der Primzahlen, zum anderen der Menge der Nicht-Primzahlen. Das heißt, was ich mache, ist, dass ich eine logische Abfrage habe, welche wahr oder falsch sein kann. Also die Abfrage, die im ursprünglichen Beispiel gemacht wurde, die zwei Kategorien, ist eben: Ist das Ganze eine Primzahl oder gehört das zur Komplementärmenge der Primzahlen? Bzw. anders formuliert, kann ich die Abfrage machen: Ist es eine Primzahl? Wenn ja, ist es eine Primzahl, und wenn nein, gehört es zur Komplementärmenge. Das heißt, ich habe eine logische Abfrage, welche wahr oder falsch sein kann. Und was jetzt dieser ifelse()-Befehl erlaubt bzw. wie die Struktur von dem ifelse()-Befehl aufgebaut ist, ist, dass an erster Stelle eine logische Abfrage steht. Und dann für die zwei Optionen, die zwei Antwortmöglichkeiten, für diese logische Abfrage, dass die logische Abfrage wahr ist auf der einen Seite, dort würde dann eine Antwortmöglichkeit, eine Output-Möglichkeit, definiert werden. Und als dritte Option hätten wir dann die Output-Möglichkeit, die erscheint, wenn diese logische Abfrage mit falsch oder false beantwortet wird.

5.2. Benötigte Mengen erstellen

Text in R

Scriptfenster	G <- 1:10 P <- c(2,3,5,7) Pc<- setdiff(G,P)
Ausgabe Konsole	> G <- 1:10 > P <- c(2,3,5,7) > Pc<- setdiff(G,P)

Sprechtext

Und für unser Illustrationsbeispiel benutze ich entsprechend diesen ifelse()-Befehl jetzt, um eine Zuordnung der Elemente der Urliste zu Primzahlen oder bei eben der Komplementärmenge zu den Primzahlen durchzuführen. Was ich jetzt dafür mache, ist, dass ich zuerst meine Grundmenge

definiere, um einfach die Mengen noch mal deutlich zu machen. Das heißt, ich würde die Grundmenge meinetwegen G nennen und für die Grundmenge an dieser Stelle für die Zahlen, die ich betrachte, ist es ausreichend, dass ich als Menge der Zahlen alle ganzen Zahlen von 1 bis 10 betrachte, was ich eben in R über diese Doppelpunktnotation kurz vermerken kann. Und das heißt, ich habe hier einmal die Grundmenge als alle ganzen Zahlen zwischen 1 und 10. Die Menge der Primzahlen, welche, wie gesagt, meine erste Ausprägungsmöglichkeit ist, wären dann in diesem Fall die Zahlen 2, 3, 5 und 7, was eben die Primzahlen zwischen 1 und 10 sind. Und die zweite Gruppe bzw. die Komplementärgruppe, die ich meinetwegen mit Pc für Komplementärmenge notieren würde, ist dann eben die Menge aller ganzen Zahlen zwischen 1 und 10, die nicht zu den Primzahlen gehören. Das kann ich auf verschiedene Arten machen. Ich könnte das direkt eingeben jetzt. Sie sollten aber auch aus der Mathe-Veranstaltung ggf. den Befehl setdiff() kennen, welcher Ihnen die Differenz zwischen zwei Mengen gibt, was dann letzten Endes an dieser Stelle, wenn die eine Menge die Grundmenge ist, eben die Komplementärmenge ist. Das heißt, ich kann hier das Ganze über setdiff() definieren, wenn P vernünftig definiert ist. Das heißt, ich würde jetzt hier diese drei Befehle nochmal ausführen und habe dann dementsprechend meine Mengen oder meine Ausprägungsmöglichkeiten definiert.

5.3. Der is.element()-Befehl

Text in R

Scriptfenster	x is.element(x,P)
Ausgabe Konsole	> x [1] 4 3 2 4 4 7 4 3 > is.element(x,P) [1] FALSE TRUE TRUE FALSE FALSE TRUE FALSE TRUE

Sprechtext

Ziel ist es letzten Endes jetzt über diesen ifelse()-Befehl die einzelnen Elemente des Beobachtungsvektors der Urliste diesen drei Ausprägungsmöglichkeiten zuzuweisen. Was ich dafür brauche, ist zum einen der ifelse()-Befehl, welchen ich jetzt entsprechend eintippe, und dann ein Befehl, welcher mir die logische Abfrage beantwortet, ob die jeweiligen Elemente aus dem Vektor der Urliste eben jetzt Primzahlen sind oder nicht. Und der Befehl, über den ich das machen kann, auch ein flexibler Befehl, den Sie kennen sollten, ist der is.element()-Befehl. Dieser fragt letzten Endes für ein Objekt, ein Vektorobjekt üblicherweise, was in unserem Fall unser Vektorobjekt x wäre, ab, ob Elemente aus diesem Vektorobjekt in den Elementen eines zweiten Vektorobjektes zu finden sind, was in unserem Fall P wäre, also die Menge der Primzahlen. Das heißt, wir fragen hier: Sind die einzelnen Elemente aus unserem Vektor x Primzahlen? Wenn ja, kriege ich bei dieser logischen Abfrage die Antwort TRUE, und wenn nein, kriege ich an dieser Stelle die Antwort FALSE. Das heißt, wenn ich hier mein x betrachte, und um das in Erinnerung zu rufen, unser x war 4, 3, 2, 4 usw. Und wenn ich jetzt diesen Befehl is.element() ausführe, dann sehe ich, dass hier eben als Antwortmöglichkeiten kommen: Für das erste Element aus x, also der 4, kriege ich die Antwort FALSE, weil 4 ist keine Primzahl bzw. ist nicht in diesem Vektor P enthalten. 3 hingegen ist eine Primzahl, das heißt, hier habe ich TRUE. 2 ist auch eine Primzahl, ist auch gleich TRUE usw. Das heißt, wenn ich gucken möchte, ob alle Elemente in x jeweils Primzahlen sind oder nicht, nutze ich eben diesen Befehl is.element(x,P), wenn ich die Primzahlen unter P abgespeichert habe, und sehe dann dadurch, dass eben für 4 das nicht der Fall ist, während das zweite Element 3 eben eine Primzahl ist usw.

5.4. ifelse()-Befehl zusammentragen

Text in R

Scriptfenster	ifelse(is.element(x,P),"P","Pc")
Ausgabe Konsole	> ifelse(is.element(x,P),"P","Pc") [1] "Pc" "P" "P" "Pc" "Pc" "P" "Pc" "P"

Sprechtext

Und diesen logischen Vektor, der durch diesen is.element()-Befehl erzeugt wird, kann ich dann nutzen, um über den ifelse()-Befehl Antworten zu generieren, wo ich eben jetzt über diesen ifelse()-Befehl zwei Ausprägungsmöglichkeiten definiere, die angegeben werden sollen. Was ich hier an dieser Stelle machen würde, ist, dass ich einfach sage, wenn das Ganze eine Primzahl ist, dann gib mir bitte den character string P wieder. Und wenn das Ganze keine Primzahl ist, dann würde ich meinetwegen eben diese Kombination Pc für Komplement der Primzahlen als Ausgabe haben. Und wenn ich diesen Befehl jetzt ausführe, dann sehe ich, dass ich anstelle des logischen Vektors FALSE TRUE usw. die Antwortoption Pc für keine Primzahl oder Komplementärmenge zu der Menge der Primzahlen, und P für Primzahlen ausgegeben kriege.

5.5. Weitere Strukturierungsmöglichkeiten: Der factor()-Befehl

Text in R

Scriptfenster	factor(ifelse(is.element(x,P),"P","Pc")) factor(ifelse(is.element(x,P),"P","Pc"),level=c("Pc","P")) factor(ifelse(is.element(x,P),"P","Pc"),level=c("Pc","P"), labels=c("noPrime","Prime")) a3 <- factor(ifelse(is.element(x,P),"P","Pc"),level=c("Pc","P"), labels=c("noPrime","Prime"))
Ausgabe Konsole	> factor(ifelse(is.element(x,P),"P","Pc")) [1] Pc P P Pc Pc P Pc P Levels: P Pc > factor(ifelse(is.element(x,P),"P","Pc"),level=c("Pc","P")) [1] Pc P P Pc Pc P Pc P Levels: Pc P > factor(ifelse(is.element(x,P),"P","Pc"),level=c("Pc","P"), labels=c("noPrime","Prime")) [1] noPrime Prime Prime noPrime noPrime Prime noPrime Prime Levels: noPrime Prime > a3 <- factor(ifelse(is.element(x,P),"P","Pc"),level=c("Pc","P"), labels=c("noPrime","Prime"))

Sprechtext

Was ich jetzt in dem letzten Schritt noch tun kann, um das Ganze ein Stück weit analog zu der oberen Struktur des cut()-Befehls zu machen, ist, dass ich das Ganze noch als Faktorobjekt definiere. Das mache ich über den factor()-Befehl, den Sie kennen sollten, und würde das Ganze dann eben ineinander verschachteln und habe dann anstelle von einem character string nun eben ein Faktorobjekt, was ich darstellen kann, wo dann eben die Level definiert sind. Ich könnte dann in einem letzten Schritt auch noch diese Level definieren, könnte bspw. eine andere Reihenfolge machen, dass ich eben sage: Ich möchte an dieser Stelle zuerst die Komplementärmenge haben, weil die enthält niedrigere Zahlen, enthält u.a. die Zahl 1, während die Primzahlen dann erst bei 2 anfangen, könnte das Ganze also dann anders strukturieren. Dann sehe ich hier, dass die Reihenfolge bei den Levels jetzt verändert wurde. Und ich könnte, wenn das gewünscht ist, notwendig ist, eben

anstatt dieser Option Pc über den factor()-Befehl auch Labels setzen. Und ich könnte hier zum Beispiel dann sagen, statt Komplementärmenge möchte ich "noPrime" ausgegeben kriegen. Prime ist englisch für Primzahl. Und als Label für die Primzahlen würde ich dann eben meinetwegen "Prime" eingeben. Oder Primzahl geht natürlich auch, wenn man das Ganze Deutsch halten möchte. Und dann würde ich das Ganze ausgeben und sehe jetzt hier, dass ich das mehr oder weniger analog habe zu dem cut()-Befehl als Faktorobjekt, wo jetzt alle Elemente aus dem Vektor meiner Urliste in der Struktur der zwei Ausprägungsmöglichkeiten Primzahl oder eben keine Primzahl dargelegt sind. Und das Ganze könnte ich dann hier an der Stelle noch mal unter a3 abspeichern, damit das dann entsprechend analog ist zu den obigen Darstellungen, und wäre damit am Ende von diesen drei R-Ausprägungsvarianten.

Folie 13 – Vielen Dank für die Aufmerksamkeit

Folientext

Inhalt und Gestaltung

- Dr. Alexander Silbersdorff

Barrierefreiheit und Gestaltung

- BaLLviHo: Dr. Nina-Kristin Meister, Katrin Lux, Thomas Finkbeiner, Kristina Schneider, Lea Dammann, Julia Berginski

Unterstützung

- Sina Ike, Miriam Panni

Abbildungen grafischer Logos

- Sign Lab Göttingen
- Zentrum für Statistik Göttingen
- Campus-Institut Data Science Göttingen
- Twillo
- Yomma
- Georg-August-Universität Göttingen

Angabe CC-Lizenz

- Folien und Videos sind unter CC BY (4.0) lizensiert - sofern nicht anderweitig angegeben.

Sprechtext

Damit bin ich am Ende dieses Videos und bedanke mich für die Aufmerksamkeit.