kleine Einführung in R

Dies ist eine kleine Einführung in die Arbeit mit R aus einem Seminar. Es ist noch nicht final für die Veröffentlichung bearbeitet / korrigiert. Sie können es aber schon als Einstieg in R nutzen.

1. Was ist R?

R ist eine Open-Source-Software (also eine Software, deren Quellcode frei verfügbar/öffentlich ist) – und eine Programmiersprache, die hauptsächlich für die Statistik benutzt wird. R kann man sowohl für die statistische Datenanalyse benutzen als auch zu der Erstellung von Grafiken.

1.1 Kleiner Einblick in die Geschichte von R

Mitte der 70er Jahre – also vor ca. 50 Jahren wurde die **Programmiersprache S** von John Chambers et al. in den USA entwickelt, um statistische Simulationen und Grafiken zu erstellen. **Ende der 80er Jahre** wurde daraus **S-PLUS** für den kommerziellen Gebrauch entwickelt. Seither wird die Software zur Datenanalyse und zur statistischen Modellierung genutzt. R ist nichts Weiteres als eine **Weiterentwicklung** der Programmiersprache, die auf **S aufbaut** und wurde Anfang der 90er Jahre initiiert. Inzwischen ist R ein weit **verbreitetes Tool** und wird in vielen Bereich benutzt.

Die Informationen sind von folgender Quelle:
https://tu-dresden.de/gsw/phil/iso/mes/ressourcen/dateien/prof/lehre/freieS/Dateien/Einfuehrung_R.pdf?lang=de

1.2 R im Vergleich zu anderen Statistikprogrammen

R

Ein großes Argument für R ist, dass R und R Studio **kostenlos** sind. Egal ob man R als einzelne*r Student*in für die Uni benutzt, oder in einem nationalen Statistikunternehmen arbeitet – R kann kostenlos runtergeladen und genutzt werden. Es gibt auch **keine Upgrade Version**, die einen dazu „anstiftet“, eine kostenpflichtige Version zu kaufen.

Vorteilhaft ist, dass du zu R sehr viele Anleitungen findest. Egal ob du Anfänger*in bist, oder Fortgeschrittene*r, meistens findest du eine Lösung zu deinem Problem. Zudem ist es egal, auf welcher Software du R runterladen willst: MacOS, Windows, Linux o.Ä.!

Wenn du keine vereinzelten Kenntnisse beim Coden mitbringst, wirst du anfangs wahrscheinlich etwas verwirrt und überfordert sein, allerdings kommt das mit der Zeit. R ist wie eine gesprochene Sprache: Du musst sie regelmäßig üben und wenn du das Programm eine Zeit lang nicht benutzt hast, kann es auch etwas dauern, wieder reinzukommen. Allerdings gewöhnst du dich recht schnell wieder daran.

Unvorteilhaft: Bei sehr umfangreichen Rechenoperationen oder Datensätzen kann das Auswerten etwas länger dauern, v.a. bei Windows und/oder, wenn das Programm nicht das Neueste Update hat. Aber v.a. für Basics an der Uni ist das Programm gut geeignet, um reinzukommen.

Stata

Stata habe ich persönlich noch nicht benutzt, aber das ist auch ein gängiges Statistikprogramm.

Vorteilhaft: auch dieses Programm ist über diverse Betriebssysteme verfügbar, egal ob MacOS, Windows oder Unix-Betriebssysteme (Linux gehört z.B. dazu).

Unvorteilhaft: Stata ist für **niemanden kostenfrei**. Das Unternehmen bietet unterschiedliche „Packages“ an, die von **ca. 57€ für Studierende (für 6 Monate)** zu ca. **1090€ für einen nicht akademischen Gebrauch (für 1 Jahr)** reichen. [In den Beschreibungen steht immer „ab“, also kann es auch noch teurer werden]. Die Lizenz muss also immer wieder **erneuert** werden. Aber: wer sich unsicher ist, ob Stata das passende Programm ist: es gibt eine 30-tägige Testversion, die kostenfrei ist.

Die Informationen sind von folgender Quelle: https://stata-germany.com/?gad_source=1&gclid=Cj0KCQjw4cS-BhDGARIsABg4_J2KjN-41gcEo7-jxPrt8qBGTcaXLUytI_Mmwnm6-KwZeZcgbFipwu8aAvrIEALw_wcB#!/Preise

SPSS

Auch SPSS habe ich persönlich noch nicht benutzt, dennoch kenne ich einige, die damit im universitären Kontext schon gearbeitet haben.

Vorteilhaft: Vor allem Menschen, die sich nicht gerne mit dem Schreiben von Codes beschäftigen, können von diesem Programm profitieren. SPSS bietet eine Drag- and-Drop- Funktionalität und kann die **Daten analysieren, ohne Codes schreiben zu müssen**, was z.B. bei R auf jeden Fall erfordert wird. Der Output kann nach persönlichen Bedürfnissen optimiert werden und geht meistens auch recht schnell. Die Syntax kann z.B. mit R oder Python ergänzt werden.

Unvorteilhaft: SPSS ist auch für jeden kostenpflichtig. Das **Basisabonnement beläuft sich auf 1188€** pro Jahr, und kann noch mit 3 Add-Ons erweitert werden, die jeweils auch ca. 950€ jährlich kosten. Studierende hingegen können SPSS bei unterschiedlichen Anbietern erwerben, und so unterschiedliche Lizenzen erwerben – so kann man für **29€ eine sechsmonatige Lizenz erwerben und für 143€ eine dreijährige Lizenz**. (Es gibt auch teurere Versionen für Studierende, das hängt davon ab, welche zusätzlichen Tools man braucht Hier kann man das nachschauen: https://studentdiscounts.com/.

Die Informationen sind von folgender Quelle: https://www.ibm.com/de-de/products/spss-statistics

Persönliche Begründung unserer Wahl: In unserem Seminar haben wir uns alle geeinigt, R zu nutzen. Einige kannten schon den Umgang damit und hatten das Programm bereits auf dem Laptop installiert. Zudem brauchte niemand das Programm zu kaufen. Im Unikontext gibt es oft die Möglichkeit, SPSS zu nutzen, das muss dann über die PC’s in der Universität laufen – mit Laptop ist man aber flexibler und wir konnten einen Raum nutzen, der nicht unbedingt andere PC’s erfordert hat.

2. Grundlagen von R

2.1 Installation von R

Damit du R nutzen kannst, musst du das Programm natürlich erstmal runterladen. Am besten lädst du zusätzlich RStudio runter. R funktioniert zwar auch ohne RStudio, allerdings vereinfacht RStudio die Arbeit mit R erheblich. Aber was ist der Unterschied? R ist die Programmiersprache für statistische Analysen (wie vorhin schon beschrieben). Wenn ihr nachher arbeitet, arbeitet ihr also mit R. RStudio hingegen ist eine Entwicklungsumgebung für R.

Der erste Schritt ist natürlich der Download von R:

Hierfür musst du auf folgende Seite: https://cran.r-project.org/ Auf der Seite werden euch Links angezeigt, auf die ihr zum Runterladen greifen könnt, je nachdem mit welchem Betriebssystem euer Gerät funktioniert. Bei mir ist das MacOS.

Wenn du da drauf klickst, werden dir unterschiedliche Informationen zu verschiedenen Versionen gegeben. Je nachdem wie neu oder alt dein Macbook ist, hat es nur Zugriff auf bestimmte MacOs Versionen und demnach musst du dich orientieren. Hier kannst du zwischen zwei Versionen unterscheiden.

[Exkurs: Dafür muss dir bewusst sein, welche Version du auf dem Laptop hast. Das findest du beim Macbook in den Systemeinstellungen raus, wenn du auf „Allgemein“ drückst.]

Dann wählst du die passende Version und das Paket wird geladen – in der Regel geht das sehr schnell und braucht nur ein paar Sekunden. Wenn es runtergeladen ist, klickst du drauf und folgendes Fenster öffnet sich:

Dem folgst du nacheinander, die Schritte sind selbsterklärend. *Wichtig: hier lädst du nur R runter!*

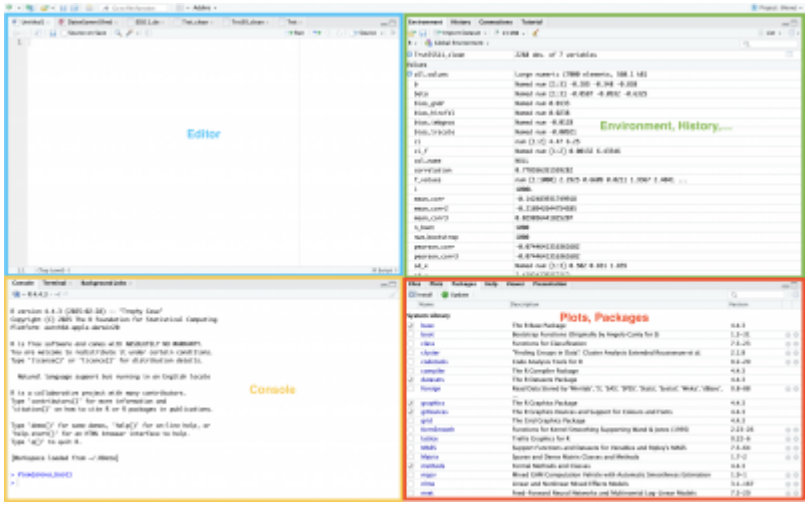
Wenn du zusätzlich RStudio runterladen willst, dann kannst du die folgende Anleitung beachten: Zuerst öffnest du diese Webseite dafür: https://posit.co/download/rstudio-desktop/ und dann klickst du auf „Install R-Studio“:

Auch das ist normalerweise recht schnell runtergeladen. Wenn du drauf klickst, öffnet sich folgendes Fenster. Zur Benutzung von R öffnest du einfach die App „RStudio“.

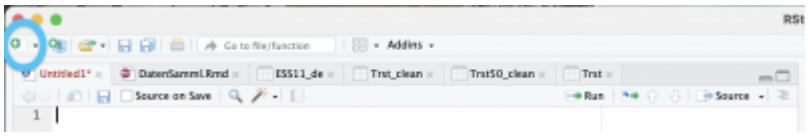
Wahrscheinlich wirst du gefragt, ob du die App wirklich öffnen willst, dann klickst du einfach auf „öffnen“. Dieses Nachfragen kann nervig erscheinen, das dient aber nur zu deinem Schutz und ist i.d.R. normal bei Apple.

2.2 Der Aufbau von R

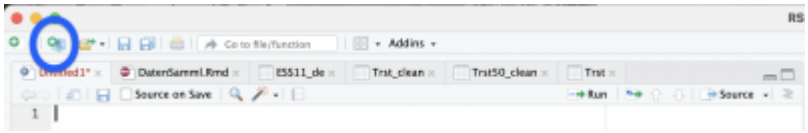
Wenn du R nun öffnest, wirst du zu Beginn vielleicht von all den Informationen und Eindrücken erschlagen. Deswegen arbeiten wir das jetzt Schritt für Schritt durch, damit du ein tiefgründiges Verständnis für das Programm aufbaust. Was du nach dem Öffnen zuerst siehst, ist das GUI: das Graphical User Interface und dieses besteht aus vier Bereichen:



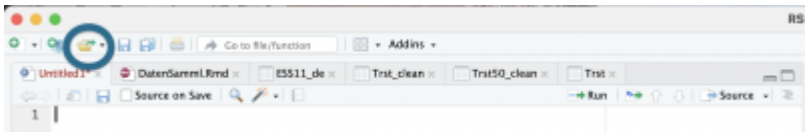
Fangen wir mit dem „Editor“ an: hiermit wirst du wohl am meisten arbeiten. Du kannst ihn dir wie ein Bereich vorstellen, indem du dein Skript erstellst. Du schreibst hier also deinen Code. Hier kannst du von unterschiedlichen Funktionen Gebrauch machen. Wenn du auf das weiße „Blatt“ mit Plus klickst, erstellst du eine neue „Seite“, in der du ein neues Skript beginnen kannst.



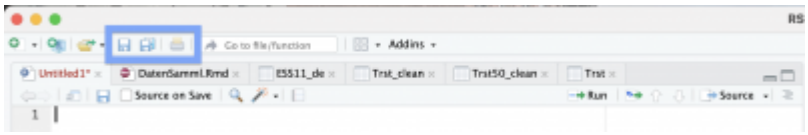
Mit dem Symbol daneben (R mit einem Plus) kannst du ein neues Projekt starten und zwischen unterschiedlichen Möglichkeiten wählen, je nachdem was du brauchst.



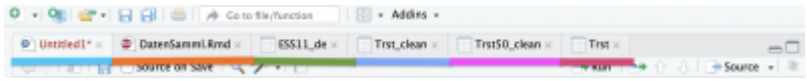
Auf das Symbol mit dem Ordner kannst du klicken, wenn du vorhandene Dateien auf deinem Laptop in R laden magst.



Die folgenden drei Symbole sind dir wahrscheinlich von anderen Systemen wie „Word“ bekannt. Die ersten beiden sind zum Speichern, mit dem anderen Symbol kannst du dein Skript drucken.



Gleich darunter werden dir die Dateien angezeigt, die du gerade offen hast. Bei mir ist das gerade ein Skript (Das Dokument „DatenSamml.rmd“, die anderen Dateien sind Tabellen (z.B. zur Übersicht, die ich über das Environment geöffnet habe). Ich kann auf jede Datei zugreifen, ohne dass der Code von einer anderen Datei verloren geht. Im Rahmen dieser Anleitung werde ich vorerst mit dem Dokument „Untitled1“ arbeiten.



Diese Datei kann natürlich auch umbenannt werden, indem du die Datei speicherst. Achte darauf, wo du deine Datei auf deinem Laptop abspeicherst, damit du sie auch immer findest, wenn du auf sie zugreifen willst.

In der „Console“ wird dir der ausgeführte Code zusammen mit den Ergebnissen angezeigt. Dort führt R also deine Befehle aus. Auch Fehlermeldungen werden dir dort angezeigt, die dir dann den Hinweis liefern, wieso und wo genau eine Fehlermeldung eingetreten ist. Mit dem „Environment“ erhälst du einen Überblick über die selbst erstellten Objekte. Es werden dir Daten, Variablen, Werte, usw. angezeigt. Im unteren rechten Bereich werden dir unterschiedliche Dateien, Plots, Hilfsangebote und eine Übersicht über die verschiedenen Pakete angezeigt, die du dir installieren kannst. Zu diesen drei Bereichen werden im Laufe der Anleitung ausführlichere Übersichten und Erklärungen kommen.

3. Erste Arbeitsschritte mit R

Jetzt geht’s ans Arbeiten mit R. Wie bereits erwähnt, findet der größte Teil der Arbeit im Editor statt. Fangen wir mal mit grundlegenden Rechnungen an. Jetzt willst du natürlich die Ergebnisse der jeweiligen Rechnungen haben. Hierfür kannst du unterschiedliche Ausführungsmöglichkeiten nutzen:

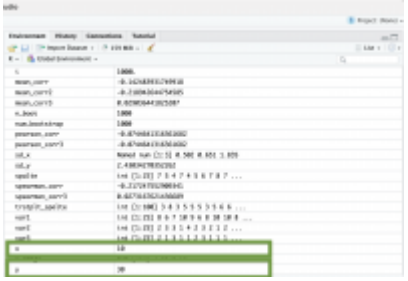
1. Du kannst oben auf „Run“ drücken (dann wird aber nur eine Zeile ausgerechnet, und zwar die, die du gerade ausgewählt hast, in diesem Falle würde nur die vierte Zeile ausgerechnet werden, also 30*80.).

2. Wenn du die Ergebnisse schneller und einfacher haben willst, kannst du auch folgenden Shortcut benutzen (dieser berechnet dann alle Zeilen): cmd+shift+enter beim Mac, strg+shift+enter bei Windows/Linux.

Wie du siehst, erscheinen die Ergebnisse unten in der „Console“.

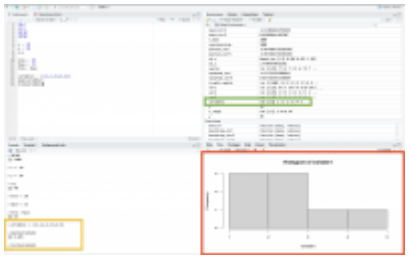
Für komplexere Geschichten in R, müssen z.B. Variablen vorerst definiert werden, damit du anschließend damit weiterrechnen kannst. Das geht, indem du einer Variable (z.B. x) einen Wert zuweist, dafür brauchst du einen Pfeil (←). Der Pfeil steht fürs Definieren (z.B. x definiert 10, y definiert 30. Also x+y=40). Das kannst du theoretisch auch mit beliebigen anderen Wörtern/Namen machen:

Im Moment, indem du die Berechnung durchläufst, ploppt im „Environment“ etwas auf. Hier kannst du dann deine Variablen einsehen – bei mir sind jetzt recht viele, die von einem Seminar sind. Wenn du nach einer Zeit den Überblick verlierst, so wie bei mir, dann kannst du diese Liste auch „säubern“, indem du oben auf den Besen klickst. Dann wird die Liste geleert. Du kannst zudem auch Funktionen oder Daten einsehen, die du im Laufe der Zeit definiert hast. Wenn du etwas genauer deine Schritte verfolgen magst, kannst du deine letzten Schritte unter „History“ aufrufen.



Definieren wir mal eine Variable und rechnen den Mittelwert aus. Zusätzlich wollen wir ein Histogramm erstellen. In der „Console“ werden dir die Ergebnisse und deine eingegebenen Definitionen angezeigt. Im „Environment“ entsteht dann deine Variable (grün umrandet), die du einsehen kannst. Wenn du beispielsweise Histogramme erstellst, werden dir die im „Plot“-Bereich

angezeigt.



3.1 Pakete

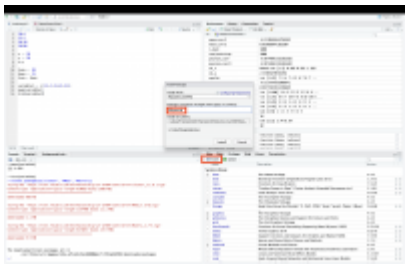
Wenn du R runterlädst, hast du das Basic Paket, mit dem du einfache statistische und mathematische Dinge machen/berechnen kannst. Je tiefgründiger du aber mit R arbeitest, desto sinnvoller und notwendiger werden zusätzliche Pakete (Packages). Keine Angst, die sind alle kostenlos und diese Pakete runterzuladen kostet meistens nur einige Sekunden. Es gibt unterschiedliche Methoden, diese runterzuladen. Am besten du entscheidest dich, je nachdem, welche und wie viele Pakete du runterladen willst.

1. Methode: hauptsächlich, wenn du nur vereinzelte Pakete runterladen willst: beispielsweise „Tidyverse“. Dieses Paket beinhaltet automatisch mehrere Pakete, die hilfreich sein können:

Paket	Beschreibung
ggplot2	Erstellen von Datenvisualisierungen mit einem "Grammar of Graphics"-Ansatz.
dplyr	Datenmanipulation, wie Filtern, Gruppieren und Transformieren von Daten.
tidyr	Datenaufbereitung durch Umstrukturierung von Daten (z.B. Pivotieren).
readr	Importieren und Lesen von rechteckigen Datendateien (CSV, TSV).
purrr	Funktionale Programmierung mit Funktionen wie map() zur Iteration.
tibble	Verbesserte Datenformen mit besserer Druckausgabe und Funktionen zur sicheren Handhabung von Daten.
stringr	String-Manipulation und Textverarbeitung.
forcats	Funktionen zur Arbeit mit kategorialen Variablen (Faktoren).

(Die Tabelle habe ich aus Chatgpt, hier der Link für die Konversation: https://chatgpt.com/share/67e14a77-eaa4-8012-bdf5-8acc8bdbf9a7)

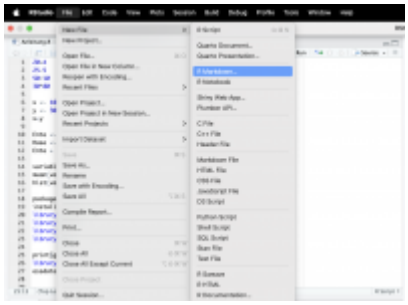
Du kannst das Paket „manuell“ runterladen, indem du unten rechts in der Abteilung „Packages“ auf „Install“ klickst. Dann öffnet sich das Fenster und du kannst dein gewünschtes Paket aufrufen. Anschließend klickst du auf „Install“.

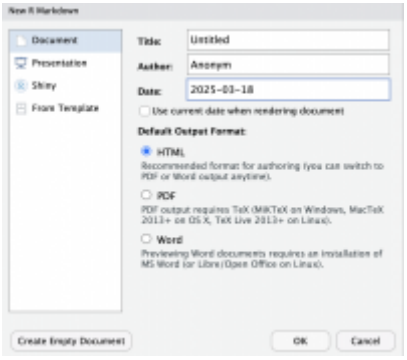


2. Methode: wenn du mehrere Packages runterladen musst, kannst du eine Variable mit den Packages erstellen und diese im anschließenden Schritt ausführen. Das sieht dann folgendermaßen aus:

3.2 Good to know

In R gibt es die Funktion eine „Markdown-Datei“ zu erstellen. Diese ermöglicht es, Codes, Grafiken usw. in einem Dokument miteinander zu vereinen. Man kann sie auch als „Cheat-Sheet“ betrachten. Wenn du auf „R Markdown…“ klickst, kannst du dem Dokument und dem Autoren/der Autorin einen Namen geben. Zudem wird das Erstelldatum gespeichert.





Eine neue Datei (.rmd) öffnet sich und darin arbeitest du am besten, v.a. wenn es um komplexere Datensätze geht. In der Übersicht wird dir die Markdown kurz erklärt und wozu sie genutzt wird. Das ist deine Übersicht in R:

3.3 Datensätze in R laden

Vor allem im universitären Kontext kann es vorkommen, dass du mit Datensätzen arbeitest, die bereits existieren. Somit musst du die Studien und Umfragen nicht unbedingt selbst durchführen, kannst aber dein Verständnis im statistischen Bereich trotzdem so sehr vertiefen wie du willst. Wir

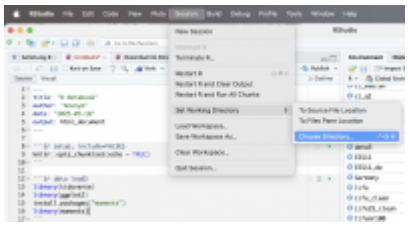
haben mit den ESS Datensätzen gearbeitet, deswegen werden diese im Fokus stehen. Wir haben mit der 11. Runde des ESS Datensatzes gearbeitet, das ist die rezenteste. Du kommst mit folgenden Schritten dahin. Zuerst rufst du die ESS Seite auf, dahin gelingst du mit folgendem Link: https://ess.sikt.no/en/datafile/242aaa39-3bbb-40f5-98bf-bfb1ce53d8ef

Mit welchem Datensatz du arbeitest, entscheidest du natürlich selbst (oder im Seminar), je nach Interesse. Dann wählst du unter den vier Möglichkeiten „ESS11- integrated file, edition 2.0 aus“. (Wichtig!: Damit du die Daten auch aufrufen/runterladen kannst, musst du eingeloggt sein, das geht ganz einfach mit einem Google Account und es ist alles kostenlos.)

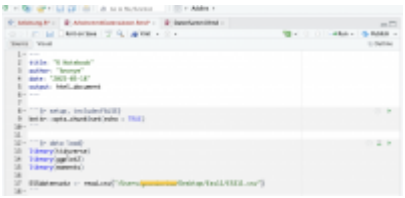
Dann hast du die Option, zwischen mehreren Programmen zu wählen, mit denen du den Datensatz runterlädst. Am einfachsten ist das SPSS (.sav) Format oder auch das Stata (.dta) Format. Aber auch das (.CSV) Format funktioniert, dafür musst du aber das Paket „readr“-Paket installiert haben. Ich habe das Format für SPSS gewählt. Das Runterladen geht recht schnell, und braucht i.d.R. nur einige Sekunden, maximal eine Minute.

Jetzt willst du natürlich den Datensatz in R laden, damit du auch damit arbeiten kannst. Dafür musst du beachten, wo du deinen geladenen Datensatz im Computer abspeicherst, am besten kennst du den „Pfad“ zu deinem Datensatz. Ansonsten könnte folgende Fehlermeldung aufleuchten und das hindert dich an der Arbeit mit dem Datensatz, denn dein Computer findet den Datensatz einfach nicht:

Tipp: Falls du es nicht weißt: Um den „Pfad“ zu finden, kannst du oben in der Leiste über die „Session“ die „Working Directory“ auswählen. Dann bekommst du Auskunft über den Pfad deiner Datei.

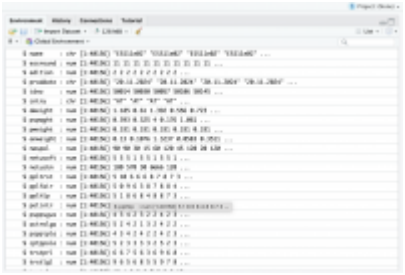


In der „Console“ wird dir dann der „Aufenthaltsort“ der Datei beschrieben. Spätestens jetzt ist alles bereit, dass du den Datensatz in R laden kannst. Zuerst überlegst du dir einen Namen, der den Datensatz definiert, z.B. „ESSdatensatz“. Das machst du mit dem Pfeil, wie du vorhin beim Definieren von verschiedenen Variablen gelernt hast. Dann soll die Datei „gelesen“ werden, die du auf deinem Laptop abgespeichert hast und die du ja wie beschrieben mit dem Pfad findest.



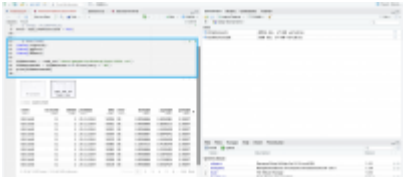
Wenn du auf den grünen Pfeil rechts klickst, durchläuft R immer den ganzen Chunk (das sind die einzelnen grauen „Bereiche“). Das kannst du aber auch ganz einfach mit einem Shortcut durchlaufen lassen, wie ich euch im Kapitel zu Shortcuts zeigen werde. Wenn der Chunk durchlaufen wurde, erscheint im „Environment“ eine neue „Data“.

Wenn du auf diese klickst, öffnet sich ein Tab mit den ganzen Angaben (also vom ganzen Datensatz, den du runtergeladen hast). Das kannst du aber auch einfach wieder zuklappen, diese Angaben dienen nur zur genaueren Information und du brauchst sie in der Regel nicht dauerhaft. Das sieht dann ungefähr so aus:

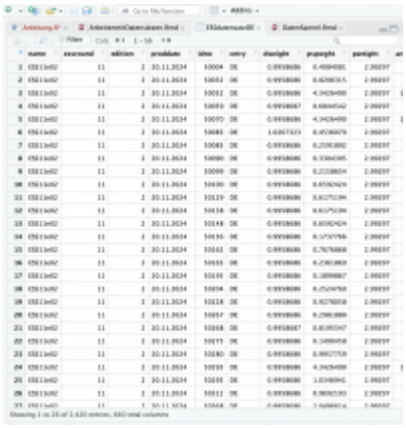


3.4 Mit Datensätzen arbeiten

Mit diesen Datensätzen kannst du natürlich viele unterschiedliche Dinge berechnen und analysieren, je nachdem was dein Fokus ist. Die Datensätze von ESS beinhalten Daten von vielen unterschiedlichen Ländern und je nachdem ist dein Fokus auf einem bestimmten Aspekt, oder einem bestimmten Land. Dann ist es sinnvoll, die Daten zu filtern, damit du ausschließlich diese analysieren kannst. Nehmen wir mal an, du willst die Daten aus Deutschland analysieren. Dafür kannst du einen Code benutzen, der aus den Datensätzen nur die Daten aus Deutschland anzeigt. Anschließend kannst du dann damit weiter analysieren.



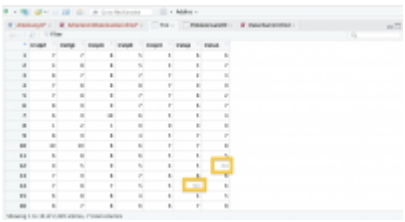
Im „Environment“ kannst du dann die Variablen dazu einsehen und wenn du darauf klickst, öffnet sich ein neuer Tab, in dem du die ganze Tabelle im Überblick hast. So sieht das dann aus (das ist natürlich nur ein Ausschnitt, also 26 von 2.460):



Ein nerviges Problem, das beim Analysieren deine Werte manipulieren kann, (indem besonders abweichende und unrealistische Werte z.B. bei der Schiefe oder Kurtosis rauskommen) sind ungültige Werte. Diese entstehen wenn bei der Umfrage (z.B. Fragen mit Antwort, bei der man eine Skala bis 10 hat) nichts oder falsche Antworten angegeben wurden. In dem Fall werden die Antworten oft mit „77“, „99“ (meistens aber mit „88“, zumindest bei mir) gekennzeichnet. Generell kann die angegebene Zahl aber nicht höher als 10 sein. Deswegen müssen diese fehlerhaften Werte ausgefiltert werden. Sagen wir, du interessierst dich für folgende Variablen: „trstprl“, „trstlgl“, „trstplc“, „trstplt“, „trstprt“, „trstep“, „trstun“ und willst aus diesen Variablen die ungültigen Werte filtern. Dafür kannst du eine Variable erstellen, die alle 7 enthält und sie definieren, wie du willst. Da es um Variablen geht, in denen das Wort „trust“ vorkommt, nennen wir die Variable jetzt einfach „Trst“:

Dann schreibst du R vor, alle Werte, die größer als 10 sind (>10) durch N.A. zu ersetzen. Du kannst auch einen anderen Code nehmen, z.B. „Trst[Trst == 77 | Trst == 88 | Trst==99] ← NA. Das ist dir selbst überlassen, im Endeffekt werden die ungültigen Werte durch NA ersetzt, was ja das Ziel ist.

Wenn du nun die Tabelle öffnest, kannst du sehen, dass einige N.A.-Werte im Datensatz vorhanden sind:



Bei so einer großen Tabelle kann es natürlich vorkommen, dass du den Überblick verlierst. Damit du einen Überblick erhältst, ob NA Werte vorhanden sind, kannst du einen weiteren Code aufrufen:

```r
Trst <- ESSdatensatzDE[, c("trstprl","trstlgl","trstplc","trstplt","trstprt","trstep","trstun")]
Trst[Trst > 10] <- NA

sapply(ESSdatensatzDE, function(x) sum(is.na(x)))
```

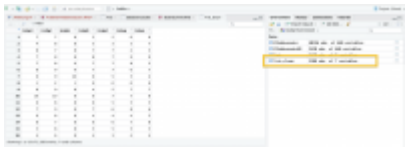
Wahrscheinlich wirst du bei der Analyse Stichproben ziehen, mit der Grundgesamtheit arbeiten ist oft für Seminarkontexte zu viel und generell nicht besonders sinnvoll. Um zu vermeiden, dass du Stichproben ziehst, die NA Werte haben, kannst du eine Variable erstellen, die von solchen Werten befreit ist. Am besten gibst du ihr einen Namen, an dem du erkennen kannst, dass sich die „gesäuberten“ Werte darin befinden. Deswegen nenne ich sie jetzt „Trst_clean“.

```r
Trst <- ESSdatensatzDE[, c("trstprl","trstlgl","trstplc","trstplt","trstprt","trstep","trstun")]
Trst[Trst > 10] <- NA

sapply(ESSdatensatzDE, function(x) sum(is.na(x)))

Trst_clean <- Trst[complete.cases(Trst), ]
```

Die Tabelle mit den sauberen Werten kannst du dann auch immer wieder im „Environment“-Bereich öffnen und einsehen.



3.5 Stichproben ziehen

Oft arbeitest du mit gezogenen Stichproben und vergleichst diese mit anderen, denn die Grundgesamtheit zu untersuchen ist sehr aufwendig und ineffizient. Von der Stichprobe schließt du dann auf die Grundgesamtheit. Wenn du dann unterschiedliche Stichprobengrößen mit anderen vergleichst, kannst du Muster und Zusammenhänge erkennen und Hypothesen testen. Das geht auch in R ganz einfach mit einem Code, der dir ermöglicht deine erwünschte Stichprobengröße zu erzeugen. Du definierst deine Stichprobe, ich habe sie jetzt „Stichprobe100“ genannt und du brauchst dafür natürlich deinen gesäuberten Datensatz, damit keine abweichenden Werte rauskommen. Deswegen habe ich den Datensatz „Trst_clean“ genommen und nicht „Trst“. Dieser Code wählt dann zufällig 100 Zeilen-frames aus dem Datensatz „Trst_clean“ aus, erstellt also eine 100er Stichprobe.

```r
Stichprobe100 <- Trst_clean [sample(nrow(Trst_clean), size = 100), ]
```

Wenn du den Code ausführst, kommt im „Environment“ eine Variable dazu: „Stichprobe100“. Auch hier kannst du drauf klicken, dann öffnet sich ein Tab, der die Tabelle mit euren 100 Stichproben anzeigt:

Wichtig: Wenn du den Code wie oben angezeigt ausführst, entstehen jedes Mal andere Stichproben! Das ist ungünstig, wenn du über längere Zeit damit arbeitest. Jedes Mal wenn du R öffnest, musst du den Code nämlich nochmal durchlaufen lassen. Es entstehen dann jedes Mal neue 100er Stichproben, was für deine Analysen manipulierend ist. Deswegen macht es Sinn, dass du den Code anpasst, das geht mit „set.seed(123)“. Das ermöglicht, dass die Stichprobenziehung beim Durchlaufen immer

identisch ist und sollte deswegen unbedingt anpegasst werden.

```r
set.seed(123)
Stichprobe100 <- Trst_clean [sample(nrow(Trst_clean), size = 100), ]
```

Tipp: Du kannst auch mehrere Stichproben gleichzeitig ziehen, du musst nicht alle Stichproben in einzelnen Schritten ziehen. Klickst du einmal auf den grünen Pfeil, entstehen gleich drei Stichproben.

```r
set.seed(123)
Stichprobe100 <- Trst_clean [sample(nrow(Trst_clean), size = 100), ]
Stichprobe250 <- Trst_clean [sample(nrow(Trst_clean), size = 250), ]
Stichprobe500 <- Trst_clean [sample(nrow(Trst_clean), size = 500), ]
```

4. R und seine unterschiedlichen Logiken

Wer sich mit R befasst und sich beispielsweise in einer Gruppe mit anderen darüber unterhält, wird bemerken, dass es in R nicht nur „den einen richtigen Weg“ gibt, um zum „Ziel“ zu kommen. Manchmal vergleichst du deine Codes mit anderen und ihr alle habt, wenn alles gut geht, die Ergebnisse, die ihr haben wolltet. Du hast aber eventuell einen anderen Weg/Code benutzt. Trotzdem funktionieren alle eure Codes. Es gibt eben unterschiedliche Logiken, die euch aber alle zum Ziel führen können.

In diesem Kapitel gehe ich auf unterschiedliche Logiken ein, die auftauchen können. In diesem Kapitel stelle ich dir drei gängige Verfahren vor und unterschiedliche Wege, wie du sie in R anwenden kannst. Es gibt in den meisten Fällen die Option, ein Verfahren durchzuführen, ohne unbedingt ein bestimmtes Paket installiert zu haben. Die Pakete vereinfachen das Coden aber in den meisten Fällen. Allerdings gilt: wenn du tiefgründigeres Verständnis für die Logik von R entwickeln willst, arbeitest du am besten mit so wenig Paketen wie möglich.

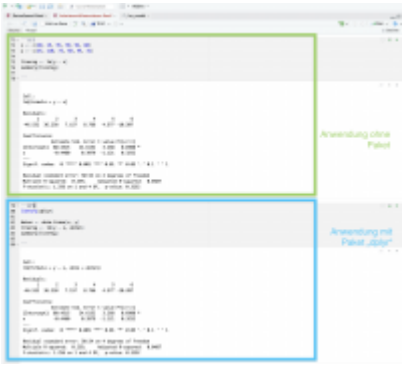
(1) Deskriptive Statistik

Die Berechnung dieser Verfahren verläuft recht einfach mit den Basisfunktionen von R. Für die Berechnung von Mittelwert, Median und Standardabweichung sind also keine Pakete notwendig. Dahingegen kannst du aber auch ein Paket anwenden, das dir diese Verfahren berechnet. So kannst du beispielsweise alternativ das „dplyr“-Paket anwenden:

Der Unterschied besteht nur daraus, dass dir bei der Variante ohne Paket mit dem Befehl „summary“ weitere Daten angegeben werden, wie z.B. das 1. Und 3. Quartil, zudem das Minimum und das Maximum. Wenn du die rausfinden willst, musst du das bei der Variante mit Paket explizit angeben. In diesem Fall brauchst du jetzt nicht unbedingt ein zusätzliches Paket, aber hier geht es auch um einfache Berechnungen.

(2) Lineare Regression

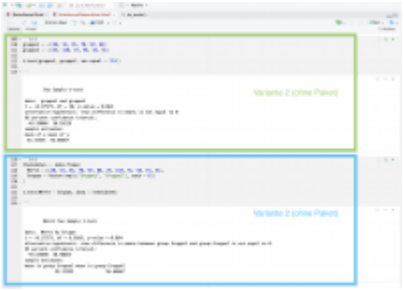
Auch eine lineare Regression kannst du mit unterschiedlichen Codes berechnen.



Die beiden Varianten unterscheiden sich in der Datenstruktur. Bei der Anwendung ohne Paket werden x und y als numerische Vektoren definiert und bei der Anwendung mit einem Paket wird ein Datenrahmen „Daten“ erstellt. Das lineare Modell wird mit der Formel „y ~ x“ und mit „Daten“ spezifiziert. Diese Methode ist allerdings eher sinnvoller für Datensätze, die etwas komplexer sind. R greift bei der ersten Variante auf die Vektoren zu, die sich im „Environment“ befinden. Bei der zweiten Variante greift R explizit auf „Daten“ zu, was die Arbeit mit komplexen Datensätzen erleichtert.

(3) T-Test

Einen T-Test kannst du auch mit verschiedener Syntax durchführen, du musst nicht mal unbedingt für die zweite Variante ein besonderes Paket installieren:



Die erste Variante führt einen T-Test durch, indem die zwei Gruppen als separate Vektoren definiert werden. Dabei trifft die Option „var.equal = True“ die Annahmen gleicher Varianzen. Bei der zweiten Variante werden die Werte von beiden Gruppen im Datenframe „ttestdaten“ zusammengefasst. Spalte „Gruppe“ markiert die Zugehörigkeit zu „Gruppe1“ und „Gruppe2“. Der T-Test wird mit der Formel Werte ~ Gruppe durchgeführt, und R vergleicht die Mittelwerte der beiden Gruppen im Datenrahmen.

Der Unterschied der Syntax in den meisten Fällen ist also, dass der Code bei der ersten Variante mit getrennten Vektoren arbeitet, während die zweite Variante die Daten in einem Datenrahmen speichert. Wie bereits erwähnt, ist diese Variante v.a. vorteilhafter, wenn du mit größeren Datensätzen arbeitest und komplexere Analysen durchführen willst. Wenn du deine Syntax mit anderen vergleichst, kann es durchaus vorkommen, dass ihr eine unterschiedliche Syntax habt. Oft ist auch erlaubt, mit KI-Systemen zu arbeiten und es kann sein, dass Programme wie chatgpt dem einen die 1. Syntax vorschlägt und dem anderen die 2. Syntax. Im Endeffekt ist es wichtig zu verstehen, was

R genau mit den Codes macht und wie das Programm damit arbeitet.

5. Shortcuts und nützliche Tools in R, die ihr kennen solltet

Shortcuts vereinfachen generell das Arbeiten mit Computern, nicht nur in R und ermöglichen ein schnelleres Arbeiten. Grundsätzlich sind folgende Shortcuts praktisch und effizient, wenn du in R schneller vorankommen willst. Deswegen solltest du diese kennen.

Nicht nur in R, aber zur Erinnerung

- Command + S –> speichert das Dokument. (Macbook) / Strg + S (Windows, Linux)

- Command + Z –> den letzten Schritt rückgängig machen (Macbook) / Strg + Z (Windows, Linux)

- Command + c –> kopieren / Command + v –> das Kopierte einfügen (Macbook) / Strg + c / Strg + v (Windows, Linux)

Spezifisch in R - Zur Erstellung eines neuen Markdown Chunks: Option + Command + I (Macbook) / Strg + Alt + I (Windows, Linux)

- Zum Ausführen des aktuellen Chunks: Shift + Command + Enter (Macbook) / Strg + Shift + Enter (Windows, Linux)

- Der Zuweisungspfeil: Option + - (Macbook) / Alt + - (Windows, Linux)

Tools um das Arbeiten mit R zu erleichtern # hinter dem Code bedeutet, dass alles dahinter nicht miteinberechnet wird. Das ist besonders am Anfang eine praktische Hilfe, weil du hinter jedem Code beschreiben kannst, was da passiert, ohne dass R bei der Berechnung verwirrt wird und eine Fehlermeldung ausgibt.

Das **„$“** bedeutet, dass du auf eine bestimmte Spalte zugreifen kannst, ohne die anderen Variablen im Code zu beachten.

Ein **Code**, den du auf **jeden Fall** kennen sollst, ist ein Code, der statistische Analysen anhand einer Schleife durchführt. Dafür erstellst du zuerst eine Liste (hier namens „spalte“), diese enthält fünf Dataframes (hier im Beispiel: Stichprobe100) und du gibst die ausgewählten Variablen ein. Die Schleife iteriert dann über die jede angegeben Spalte in der Liste, und das nur mit einem Mal ausführen. So sparst du bei einer hohen Anzahl von Spalten enorm viel Zeit, weil alles nacheinander durchgerechnet wird. Bei diesem Code wird dann mit jeder Spalte, jede einzelne angegebenes Verfahren durchlaufen, also z.B. der Mittelwert, der Median, usw. Das **„print“** gibt eine Textart aus, das die Ausgabe für jede Spalte kennzeichnet. So etwa sieht die Ausgabe dann aus:

Empfehlung für weiteres Lernen von und mit R:

Folgender Blog ist empfehlenswert: https://r4ds.hadley.nz/

Spezifischere Fragen/Probleme: https://blog.r-project.org/

Auch wenn nicht immer 100% Verlass ist auf die KI (immer überprüfen): https://chatgpt.com/