Einführung und Setup

R für empirische Wissenschaften v1.0.3

Jan Philipp Nolte

Inhaltsverzeichnis

1 Für wen ist dieses Tutorial? **2**

2 Aufbau **2**

3 Set up your lab **3**
3.1 Programmiersprache R 3
3.2 Programmierumgebung RStudio 3
3.2.1 Aufbau von RStudio 4
3.2.2 Never save your workspace 5
3.2.3 Default text encoding 5

4 Was ist ein Package? **5**
4.1 Packages installieren und laden 5
4.2 Installieren der notwendigen Packages 6

5 Update R zur neuesten Version **6**

6 Wo bekomme ich Hilfe? **6**

7 Datensätze laden **7**

1 Für wen ist dieses Tutorial?

Dieses Tutorial setzt keine Vorkenntnisse in R voraus. Somit ist es für jeden geeignet, der R lernen möchte. Aber auch jene, die bereits Vorkenntnisse mit Base R haben und in die Weiten des `tidyverse` aufbrechen wollen, kommen auf ihre Kosten. Dabei wird ein besonderer Fokus auf Probleme gelegt, mit denen empirische Wissenschaftler konfrontiert werden.

2 Aufbau

Das Tutorial ist in 6 Teile gegliedert, die sich in ihrem Umfang unterscheiden.

1. **Setup** (Installieren von R und RStudio, wichtige Anpassungen von RStudio, Packages, Wo bekomme ich Hilfe?)
2. **Datenvorbereitung** (Projektorientierung, Import/Export, Zuweisungspfeil, Datentypen, Ansatz und Funktionen des `tidyverse`, Fehler- und Warnmeldungen, Übungen)
3. **Deskriptive Statistik** (Deskriptive Maße, Umgang mit fehlenden Werten)
4. **Visualisierungen** (Säulendiagramm, Streudiagramm, Boxplot, Histogramm, Liniendiagramm (mit ANOVA Plot), Speichern)
5. **Inferenzstatistik** (Stichprobenplanung und Power, Voraussetzungen prüfen, Mittelwertsvergleich, Regressionsanalyse, p-Wert Korrektur, Varianzanalysen, Post-Hoc-Tests, Korrelationskoeffizienten, Kontigenztafeln, Explorative Faktorenanalyse)
6. **Extras** (Datenstrukturen (Unterschiede, Zugriff, Umwandlung), Funktionales Programmieren (Funktionen erstellen, Mappen, Nesten), Transformation einer Ergebnismatrix)

Nach dem Setup hast Du bereits eine voll funktionsfähige Arbeitsumgebung und bist bereit mit R durchzustarten. Das zweite Kapitel ist der mit Abstand wichtigste Abschnitt. Jeder, der schon mehrfach mit Daten gearbeitet hat, weiß, dass die Datenvorbereitung dabei die Meiste Zeit in Anspruch nimmt. Das spätere Auswerten und das zeichnen von Graphen funktioniert immer mit wenigen kurzen Befehlen, die dem selben Schema folgen. Dich auf alle Gefahren und Hürden eines unsauberen, unstrukturierten Datensatzes vorzubereiten ist Ziel von Teil 2. Da dies der umfangreichste und auch schwierigste Abschnitt ist, werden Dir zwei Übungsblöcke in einer interaktiven Lernumgebung zur Verfügung gestellt. Diese verlangen von Dir das Anwenden des gelernten auf zwei andere Datensätze (Transfer!). Abschnitt drei bringt dir die wichtigsten deskriptiven Maße und den Umgang mit fehlenden Werten bei. In Teil 4 lernst Du Funktionen kennen, Graphiken zu zeichnen, die Excel Anwender vor Neid erblassen lassen. Der 5. Abschnitt beschäftigt sich mit induktiver Statistik. Der letzte Abschnitt dient zum Einen als Ausblick auf die großartigen Dinge, die ein fortgeschrittener R Nutzer programmieren kann und zum Anderen werden einige grundlegende Konzepte in R wie Datenstrukturen genauer erläutert. Es wird empfohlen, die Reihenfolge beim Bearbeiten einzuhalten. Auch wenn Du R bereits installiert hast, solltest Du Dir das Setup durchlesen.

Falls Probleme auftreten oder Dir im Laufe des Tutorials etwas auffällt, was unklar formuliert ist oder gar fehlt, ist es erwünscht, auf Github eine neue `Issue` zu eröffnen.

Programmieren lernen ist anders als Fremdsprachen lernen. Man lernt nicht zuerst alle

Vokabeln und die Grammatik auswendig. Viel mehr lernt man nur die Grammatik in Form von grundlegenden Regeln und die wichtigsten Vokabeln. Alles andere schlägt man anwendungsbezogen immer wieder nach, bis man es auswendig kann. Es ist also keine Schmach für jede aufkommende Frage während der Übungen oder bei eigenen Projekten das Tutorial zur Hilfe zu verwenden. Im Gegenteil! Wichtig ist nur, dass man genau weiß, an welcher Stelle man suchen muss. Viel Spaß!

3 Set up your lab

Bevor wir loslegen können, müssen wir erst einmal alles Notwendige installieren. Dabei muss zwischen der eigentlichen Programmiersprache namens R und der Programmierungebung RStudio, in der Du in der gesamten Zeit arbeiten wirst, unterschieden werden.

3.1 Programmiersprache R

Windows:

- Gehe auf CRAN und wähle `Download R for Windows`
- Klick auf den Link `base`
- Dann auf `Download`.

macOS:

- Gehe auf CRAN und wähle `Download R for (Mac) OS X`.
- Klicke anschließend auf das aktuellste `.pkg` (z.B. `R-3.5.1.pkg`).

Linux:

- Füge das entsprechende Repository für deine Linux Variante in `/etc/apt/sources.list` hinzu. Für 16.04 benutze

```
deb https://cloud.r-project.org/bin/linux/ubuntu xenial-cran35/
```

Für andere Versionen xenial zu beispielsweise bionic oder trusty ändern.

- Führe anschließend die beiden Zeilen im Terminal aus.

```
sudo apt-get update
sudo apt-get install r-base r-base-dev
```

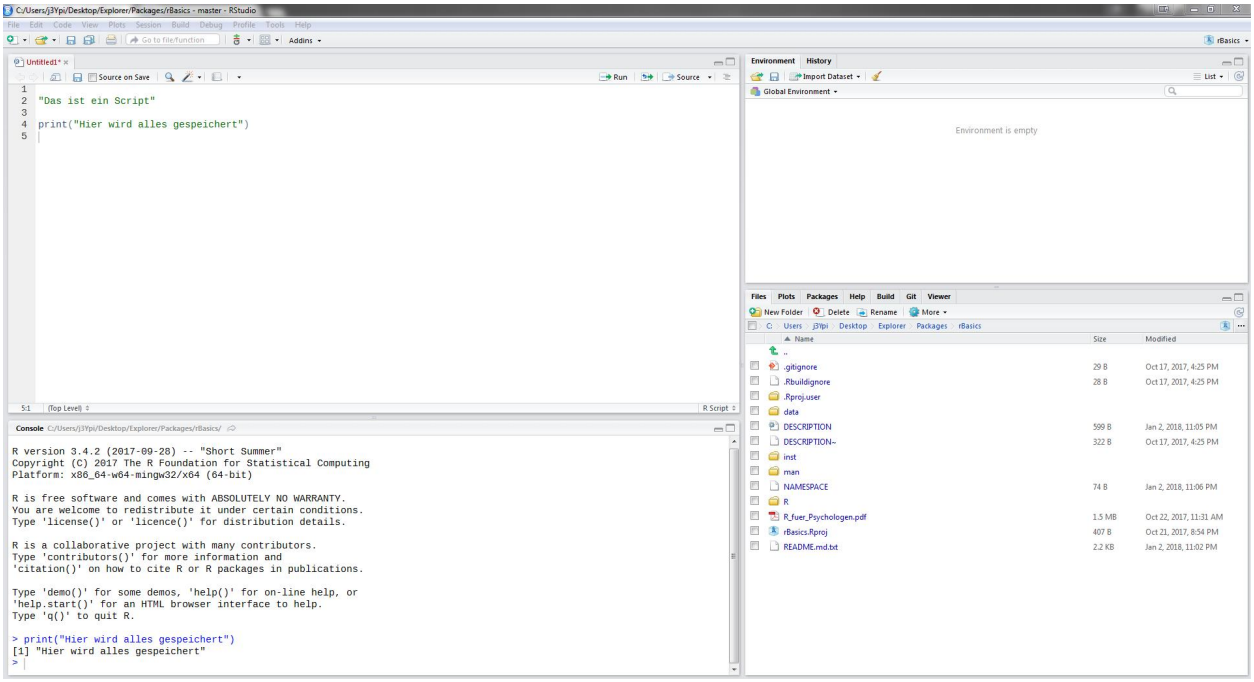
3.2 Programmierumgebung RStudio

R ist die Programmiersprache. Um komfortable mit einer Programmiersprache arbeiten zu können, braucht man allerdings auch eine *Umgebung* (auch IDE = Integrated Development Environment) genannt. Ohne ist programmieren oft mühsam. Eine Programmierumgebung

für R heißt RStudio. Wenn R installiert ist, kann man unter folgendem Link die Programmierungebung RStudio herunterladen. Drücke dazu auf den Download Button für RStudio Desktop (Open Source License) und Downloade die für dein Betriebssystem richtige Version.

3.2.1 Aufbau von RStudio

Die Oberfläche von RStudio ist in vier Teile unterteilt (siehe Abbildung 1). Unten links befindet sich die Konsole. Dort kannst du eine Zeile Code nur einmalig ausführen. Wenn man später erneut den selben Befehl ausführen möchte, muss man ihn neu eingeben. Oben links befindet sich das so genannte Script. Du kannst mehrere Scripts gleichzeitig geöffnet haben. Der große Unterschied zur Konsole ist das Speichern jeder Zeile deines Codes, sodass du später lediglich erneut das Script ausführen musst, um genau dieselben Ergebnisse erneut zu erhalten. Der vom Script erzeugte Output wird in der Konsole angezeigt. Man kann zwischen Script und Konsole mit `crtl (strg) + 1 / 2` wechseln. Um ein neues Script zu öffnen, drücke `crtl (strg) + shift + N`. Zum Ausführen drücke, in der jeweiligen Zeile `crtl (strg) + enter`. Falls alles ausgeführt werden soll, drücke zuerst `crtl (strg) + A` und führe es dann aus. Gespeichert wird mit `crtl (strg) + S`. Oben rechts befindet sich die `Environment`. In dieser kannst du sämtliche Datensätze und Variablen betrachten, die du erstellt hast. Unten rechts sind verschiedene Reiter abgebildet. Die wichtigsten für Dich sind `Plots` und `Help`. In `Plots` werden Dir wenig überraschend die erstellten Graphen und unter `Help` die Hilfen angezeigt. Wie letztere aufgerufen werden, wird unter Wo bekomme ich Hilfe? erklärt.



3.2.2 Never save your workspace

Um zu gewährleisten, dass Dein Code nicht nur auf deinem Laptop funktioniert, ist es dringend notwendig das ständige Speichern des Workspaces auszustellen.

1. Gehe zu: `Tools/Global Options.../General`
2. Entferne den Haken bei: `restore .RData into workspace at startup`
3. Ändere `Save workspace to .RData on exit` zu `never`

Keine Sorge, auf das Speichern Deines Codes hat das keine Auswirkung.

3.2.3 Default text encoding

Damit man den geschriebenen Code fehlerfrei auf anderen Geräten lesen kann, ist es wichtig dieselbe Zeichencodierung zu wählen. Die modernste ist UTF-8.

1. Gehe zu: `Tools/Global Options.../Code/Saving`
2. Ändere: `Default text encoding` zu `UTF-8`

4 Was ist ein Package?

R bietet von Beginn an eine Bandbreite von Funktionen. Da es R nun seit vielen Jahren gibt, haben sich viele fähige Programmierer an die Arbeit gemacht, Funktionen zu schreiben, die die Funktionen von `Base R` erweitern. Packages sind kostenlose Erweiterungen von Dritten, die verschiedenste Aufgaben erheblich erleichtern können. Die Packages werden auf CRAN, auf Github oder auf Bioconductor geteilt. Es gibt mittlerweile mehr als 11 000 Packages alleine auf CRAN.

4.1 Packages installieren und laden

Um ein Package zu installieren, muss man `install.packages("packageName", dependencies = TRUE)` ausführen. Wichtig ist hierbei, dass der Packagename in Anführungszeichen geschrieben ist. Das Argument `dependencies = TRUE` installiert hierbei zusätzlich alle Packages, auf denen das gewünschte Package zugreift.

```
install.packages("packageName", dependencies = TRUE)
```

Damit man auf die Funktionen des Packages zurückgreifen kann, muss das Package jedes mal aus der Bibliothek mithilfe von `library()` geladen werden. Hierbei sind keine Anführungszeichen notwendig.

```
library(packageName)
```

4.2 Installieren der notwendigen Packages

Zuerst installieren wir mit dem Befehl alle notwendigen Packages auf Deinem Computer. Wofür jedes einzelnen davon zuständig ist, wirst du im Verlaufe des Tutorials erfahren. Dies kann einige Minuten in Anspruch nehmen.

```
install.packages(c("tidyverse", "here", "rio", "broom", "car", "skimr",
                   "ggpubr", "sjstats", "devtools", "learnr", "installr"),
                 dependencies = TRUE)
```

Um auf die Übungsdatensätze und das interaktive Tutorial zugreifen zu können, musst Du das Package `rBasics` installieren. Allerdings befindet es sich nicht auf CRAN sondern Github. Der Befehl lautet daher etwas anders.

```
devtools::install_github("j3ypi/rBasics")
```

5 Update R zur neuesten Version

Für Windows und natürlich sowieso für Linux gibt es jeweiles eine einfache Möglichkeit die Version von R auf den neuesten Stand zu bringen. Für Windows brauchen wir dafür das Package `installr`. Allerdings es empfohlen den Updateprozess innerhalb von der RGui nicht von RStudio zu starten. Laden wir zuerst das Package.

```
library(installr)
```

Anschließend updaten wir R mit nur einem Befehl. Während der Installation wird man unter anderem gefragt, ob man seine Packages mit in die neue Version nehmen will - unbedingt zustimmen!

```
updateR()
```

In Linux updatest Du R wie jedes andere Programm mit

```
sudo apt-get update && sudo apt-get upgrade
```

6 Wo bekomme ich Hilfe?

Was erst einmal nach Werbung für eine Selbsthilfegruppe in der U-Bahn klingt, ist das wohl stärkste Argument für Programmiersprachen im Gegensatz zu Programmen wie SPSS - die **Community**! Das schöne an Programmiersprachen ist, dass in der Regel irgendjemand schon einmal genau dasselbe Problem gehabt hat, wie man selbst. Wenn man nach einer Fehlermeldung oder spezifischen Frage im Internet sucht, ist meist der erste Vorschlag das Forum Stackoverflow. Sollte noch niemand genau das selbe Problem gehabt haben, kann man auf Stackoverflow auch selbst eine Frage stellen. Die extrem zuvorkommende Community

antwortet erfahrungsgemäß innerhalb von wenigen Stunden. Häufig hat man aber auch bereits nach rund zehn Minuten eine Antwort von einem der fünf Millionen aktiven Forenmitgliedern. Außerdem gibt es für jedes Package und jede Funktion eine Dokumentation auf CRAN. Diese kann man entweder mit `F1` bei Öffnen der Funktion, per `?funktionsName` (z.B. ?mean) oder über die PDF Dokumentationen im Internet aufrufen.

7 Datensätze laden

Du möchtest das Gelernte sofort selbst anwenden und ausprobieren? Lade die Datensätze mit `data()` einfach aus dem Package zum Tutorial.

```
data(tipp_wm, package = "rBasics")
data(tipp_wm_wide, package = "rBasics")
data(big_five, package = "rBasics")
data(indonesisch, package = "rBasics")
data(repeated, package = "rBasics")
```