Datenanalysen mit R und der Google Analytics API

Datenanalysen mit R und der Google Analytics API – websiteboosting.com

Das Programm bzw. die Programmiersprache R erfährt in den letzten Jahren eine steigende Nachfrage. Neben dem erfreulichen Aspekt, dass es Open Source und damit komplett kostenfrei ist, kann R durch Pakete flexibel erweitert werden. Neben klassischen Datamining-Paketen und fortgeschrittenen…
Das Programm bzw. die Programmiersprache R erfährt in den letzten Jahren eine steigende Nachfrage. Neben dem erfreulichen Aspekt, dass es Open Source und damit komplett kostenfrei ist, kann R durch Pakete flexibel erweitert werden. Neben klassischen Datamining-Paketen und fortgeschrittenen Analysemethoden wie neuronalen Netzen existieren diverse Möglichkeiten, den Google-Kosmos mit R anzusprechen. Die Anbindung der API von Google Analytics, Search Console bzw. der komplette Zugriff auf Sheets/Docs sind exemplarische Beispiele hierfür. Daten können in beliebiger Kombination abgefragt, aufbereitet, analysiert und visualisiert/exportiert werden.

Analysen mit R – „flexibel wie ein Schweizer Taschenmesser“

R wird seit vielen Jahren durch eine große Community kontinuierlich weiterentwickelt. Vergleichbar mit Add-ins bei Excel können in R weitere Funktionen und Schnittstellen zum Standardsystem hinzugefügt werden. So schön diese Flexibilität und Leistungsstärke klingt (und tatsächlich ist), so unspektakulär ist die Nutzerfreundlichkeit des Systems: Eingabe von Programmcode anstelle klickbarer Icons und Mausklicks à la Excel. Was jetzt als Nachteil klingt, ist gleichzeitig ein Riesenvorteil: Sofern der Programmcode erstellt ist, kann dieser jederzeit „abgespielt“ werden, die entsprechenden Analysen/Visualisierungen dauern wenige Sekunden/Minuten und können unternehmensweit geteilt werden.

Jeder Schritt ist transparent und durch den Einsatz von Variablen voll flexibel – einmalig die ID des Google Analytics-Kontos ausgeschaut und alle Analysen können automatisch von Neuem starten. R ist quasi ein „großes Excel-Makro“, das die Daten auf Befehl Schritt für Schritt verarbeitet.

Die Software R kann unter www.r-project.org/ kostenfrei heruntergeladen werden und ist nach der Installation sofort einsatzbereit. Es empfiehlt sich, zusätzlich RStudio zu installieren (www.rstudio.com/products/rstudio/download/), welches einen deutlichen Mehrwert in der Bedienung des Systems bietet (Abb. 1).

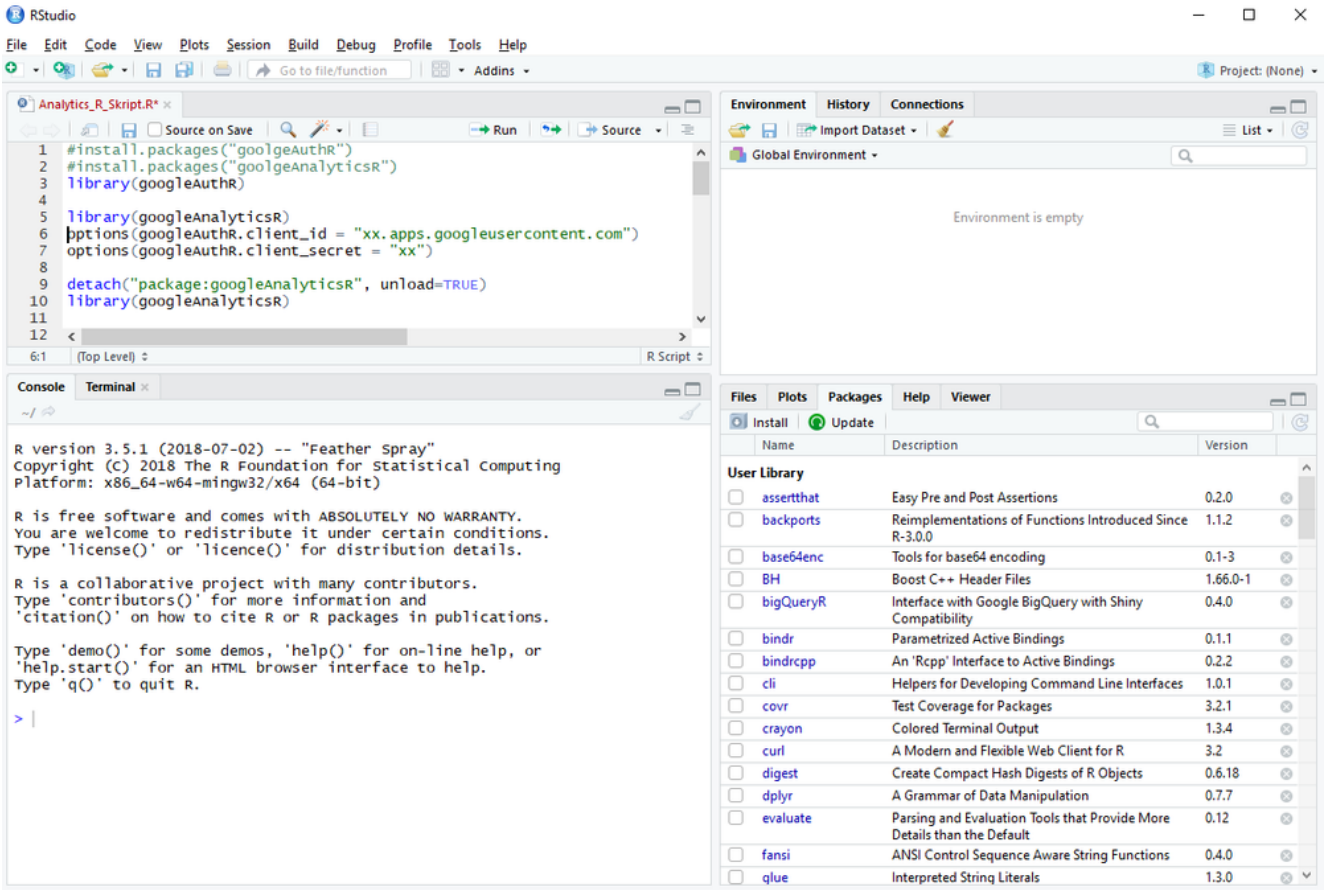


Abb. 1: RStudio „bettet“ die Software R benutzerfreundlich ein und macht die Bedienung komfortabler

Prinzipiell funktioniert R mittels Eingabe und sequenzieller Verarbeitung von Programmcode. Eingaben können sowohl in Form klassischer Rechenoperationen erfolgen (bspw. 3+3) als auch als Definition von Formeln, Funktionen, Variablen, Vektoren und Dataframes. Der Vorteil der Nutzung von Variablen ist,

dass diese nach Definition zur gesamten Laufzeit zur Verfügung stehen und im weiteren Verlauf darauf zurückgegriffen werden kann. Füllt man bspw. die Variable „ga_id“ (Variablennamen sind frei wählbar) mit der Datenansichts-ID von Google Analytics und verweist im weiteren Verlauf der Berechnungen auf die Variable anstelle einer fix definierten Nummer, kann die identische Analyse durch Neudefinition der Variable flexibilisiert werden. Variablen werden in R mittels der Syntax „Variablenname <- Variableninhalt“ definiert.

googleAnalyticsR – das flexible R-Paket

Damit R auf die API von Google Analytics zugreifen kann, muss ein Paket installiert werden (genereller Befehl: install.packages(„Paketname“). Die Installation des Pakets erfolgt nach Befehlseingabe binnen Sekunden, da die Pakete automatisch aus dem Internet geladen werden. Die Erweiterung „dependencies = TRUE“ bewirkt, dass passende Pakete zusätzlich installiert werden, insb. googleAuthR zur OAuth-Authentifizierung des Google-Nutzerkontos mit R (siehe Abb. 2).

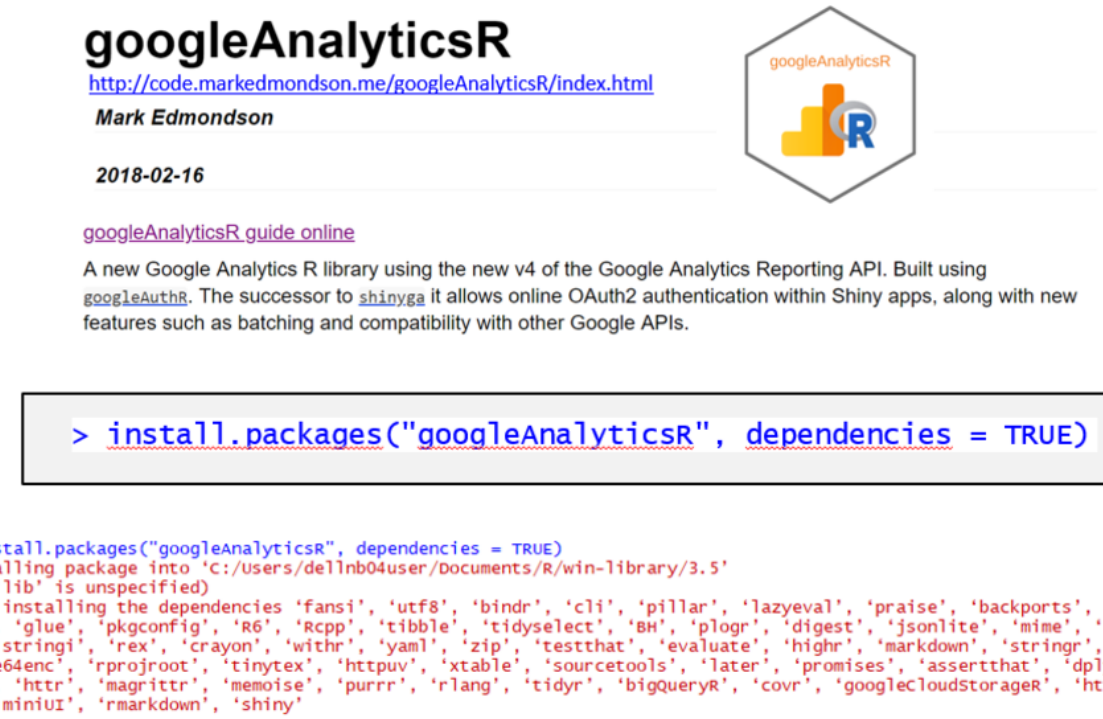


Abb. 2: Einfache Installation von googleAnalyticsR nebst weiteren Paketen direkt in R

Nachdem die gewünschten Pakete installiert wurden, stehen diese prinzipiell zur Verfügung. Damit sie tatsächlich auch genutzt werden können, müssen sie mit dem Befehl library(„Paketname“) geladen werden. Das heißt, nach Absetzen des Befehls „library(“googleAnalyticsR“)“ besteht eine Verbindung von R zur Google Analytics API. Der Befehl „ga_auth()“ bzw. „gar_auth()“ aus dem Paket googleAuthR ermöglicht die Authentifizierung von R mit dem jeweiligen Google-Konto über das OAuth-Verfahren. Ein erstmaliger Aufruf leitet zum Google-Konto-Log-in nebst Angabe der übergebenen Berechtigungen (siehe Abb. 3).

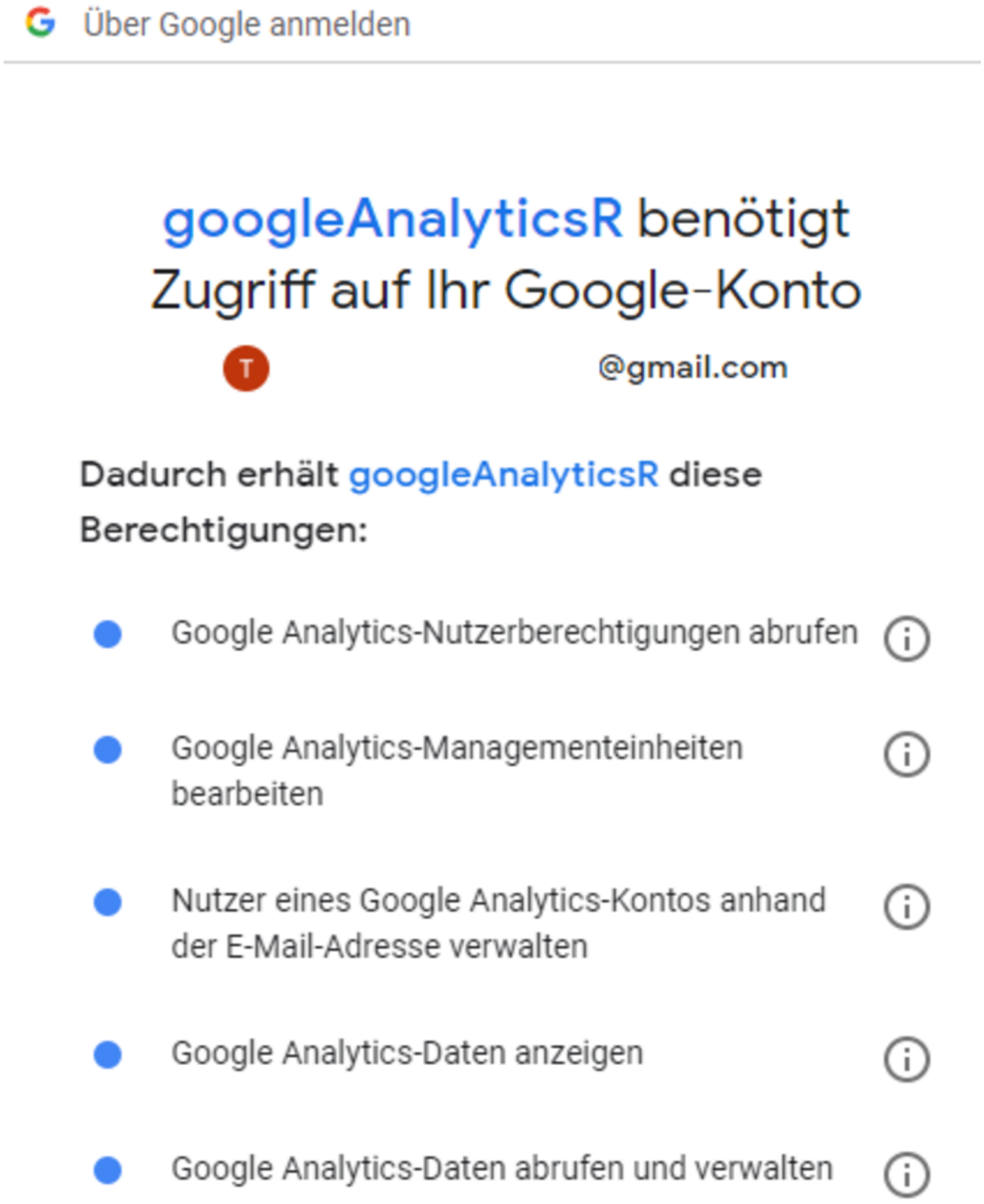


Abb. 3: Zugriff des R-Pakets auf Google-Konto

Sofern das Google-Konto bei Google Analytics mindestens das Recht „Bearbeiten“ auf eine Property hat, kann im weiteren Verlauf direkt die Google API genutzt werden. Wie angesprochen ist es sinnvoll, mit Variablen zu arbeiten und die gewünschte

Auswahl an Metriken und Dimensionen in Vektoren zu speichern. Der Befehl „ga_account_list()“ erzeugt eine Übersicht aller Analytics Propertys nebst Datenansichten (Views) im jeweiligen Google-Konto. Als Beispiel kann eine Variable „account_list“ mit genau dieser Übersicht befüllt (Befehl account_list <- ga_account_list()) und anschließend aufgerufen werden (siehe Abb. 4). Auf Basis dieser Liste kann bspw. eine Variable „ga_id“ dynamisch mit der zweiten Datenansicht befüllt werden. Hierzu wird das Feld viewId mit dem Dollarzeichen ($) aus account_list angesprochen und die zweite Position adressiert. Alternativ könnte die Variable ga_id selbstverständlich direkt mit der jeweiligen View-Nummer befüllt werden.

```
> ga_auth()
Waiting for authentication in browser...
Press Esc/Ctrl + C to abort
Authentication complete.
Token cache file: .httr-oauth
> account_list <- ga_account_list()
> account_list
  accountId   accountName internalWebPropertyId    level                  websiteUrl webPropertyId
1 [redacted]
2 [redacted]
   webPropertyName type   viewId                    viewName
1 [redacted]
2 [redacted]
> ga_id <- account_list$viewId[2]
> ga_id
[1] "65785    "
> google_analytics(ga_id,
+                  date_range = c("2018-10-01", "2018-10-30"),
+                  metrics = c("sessions", "bounces", "bounceRate"),
+                  dimensions = c("date","source", "medium"))
2018-11-15 06:31:51> Downloaded [33] rows from a total of [33].
         date              source   medium sessions bounces bounceRate
1  2018-10-01              google  organic        1       0          0
2  2018-10-02              google  organic        1       1        100
3  2018-10-03                bing  organic        1       0          0
4  2018-10-04              google  organic        1       0          0
```

Abb. 4: R-Befehle zum Abruf von R

Mit dem Befehl „google_analytics“ wird die API angesprochen und die entsprechenden Werte werden in R ausgegeben. Sofern bspw. der Datumbereich sowie die gewünschten Metriken und Dimensionen nebst der View-ID im Befehl google_analytics übergeben werden, werden die Daten unmittelbar abgerufen. Aufgrund des Abrufs der Daten über die API kann ggf. das Sampling von Daten umgangen werden. Hierzu bedarf es beim Aufruf des Befehls google_analytics der Ergänzung um den Parameter anti_sample = TRUE. Die Funktion teilt dabei den API-Aufruf in mehrere Teilaufrufe auf, damit möglichst ungefilterte Daten über die API abgerufen werden. Weiterhin

kann mit dem Parameter max = „Zahl“ die maximale Anzahl von abzurufenden Daten limitiert werden (siehe Abb. 5).

Tipp: Es besteht für Google Analytics ein sog. „Dimensionen- und Metriken-Explorer“, der anzeigt, welche Dimensionen und Metriken generell kombiniert werden dürfen und wie die entsprechenden Felder in der API heißen (bspw. ga:bounceRate für die Absprungrate in %): einfach.st/gareferenzen.

Das Paket googleAnalyticsR nutzt ein allgemeines Kontingent an API-Aufrufen von Google Analytics. Bei intensiver Nutzung und zur Sicherstellung der Verfügbarkeit sollte ein eigenes API-Kontingent über Google Developers beantragt werden (https://console.developers.google.com). Nach Registrierung bei Developers stehen diverse Google APIs zur Verfügung. Als Anpassung in R müssen vor der Authentifizierung die Client-ID sowie der geheime Clientschlüssel als Option angegeben werden (options(googleAuthR.client_id = „xx.apps.googleusercontent.com“) sowie options(googleAuthR.client_secret = „xx“). Beide Informationen sind nach der Registrierung und Aktivierung der Analytics API abrufbar. Bei Google Analytics bietet die API 50.000 Abfragen pro Tag/User bzw. alle 100 Sekunden 100 Abfragen.

Flexibler Export und Datenvisualisierungen mit ggplot2

Sofern das Ergebnis des Abrufs mittels „google_analytics“ in eine Variable/Dataframe gespeichert wird, können neben der Durchführung statistischer Auswertungen (bspw. summary(„Variable“) die Daten in ein spezifisches Dateiformat gespeichert und dann mit anderen Programmen weiterverarbeitet werden. Eine CSV-Datei wird über den Befehl „write.csv“ erzeugt (siehe Abb. 5).

Tipp: Perfekt harmoniert R auch mit Google Sheets, was damit mittelbar eine nachgelagerte Datenvisualisierung mit Google

Data Studio ermöglicht. Das Szenario wäre folgendes: Daten werden aus Google Analytics und anderen Quellen in R analysiert und verarbeitet, automatisch über das R-Paket „googlesheets“ in ein Google Spreadsheet übertragen und anschließend über Data Studio visualisiert und ggf. mit weiteren Datenquellen ergänzt.

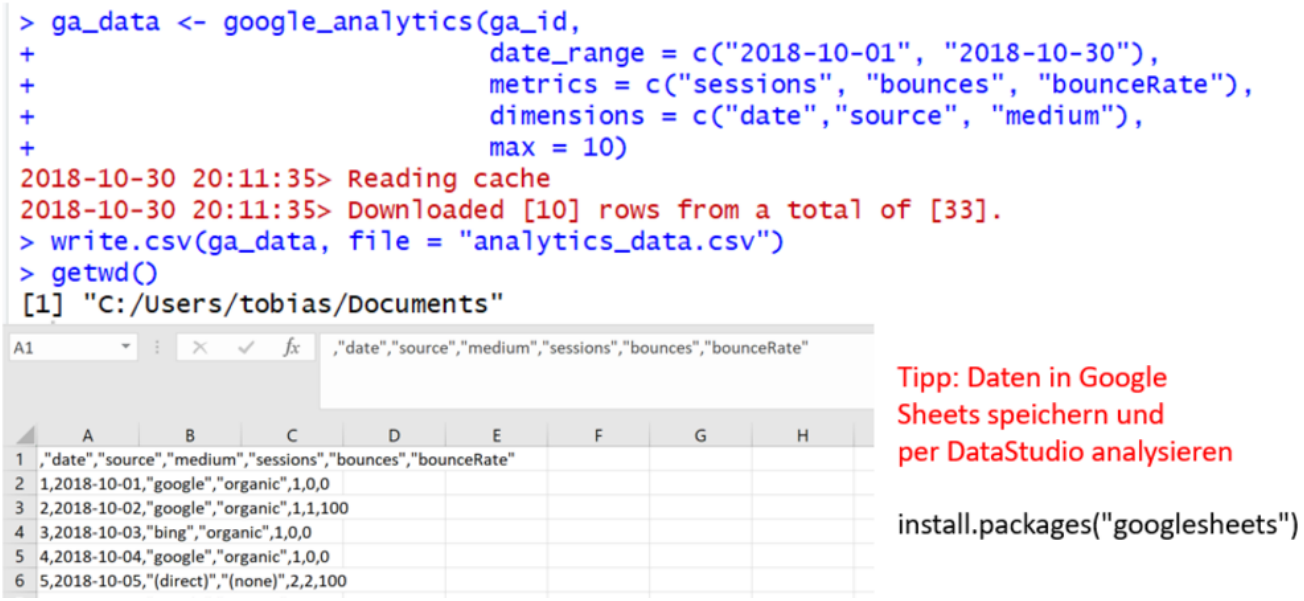


Abb. 5: Daten aus der API in R analysieren und als .csv speichern

R hat herausragende Möglichkeiten der Datenvisualisierung. Es existiert eine Vielzahl an Paketen, welche die Daten mit weiteren Analysen grafisch anreichern. Pakete wie „RColorBrewer“ erzeugen u. a. Heatmaps als Datenvisualisierung. Das Paket „ggplot2“ (install.packages(“ggplot2“) sowie Aktivierung über library(“ggplot2“)) bietet Schaubilder, die in Excel nicht existent sind. Mittels des Befehls geom_boxplot wird bspw. ein Boxplot erstellt, geom_line / geom_smooth erzeugt ein Liniendiagramm nebst nicht linearer Trendlinie (siehe Abb. 6).

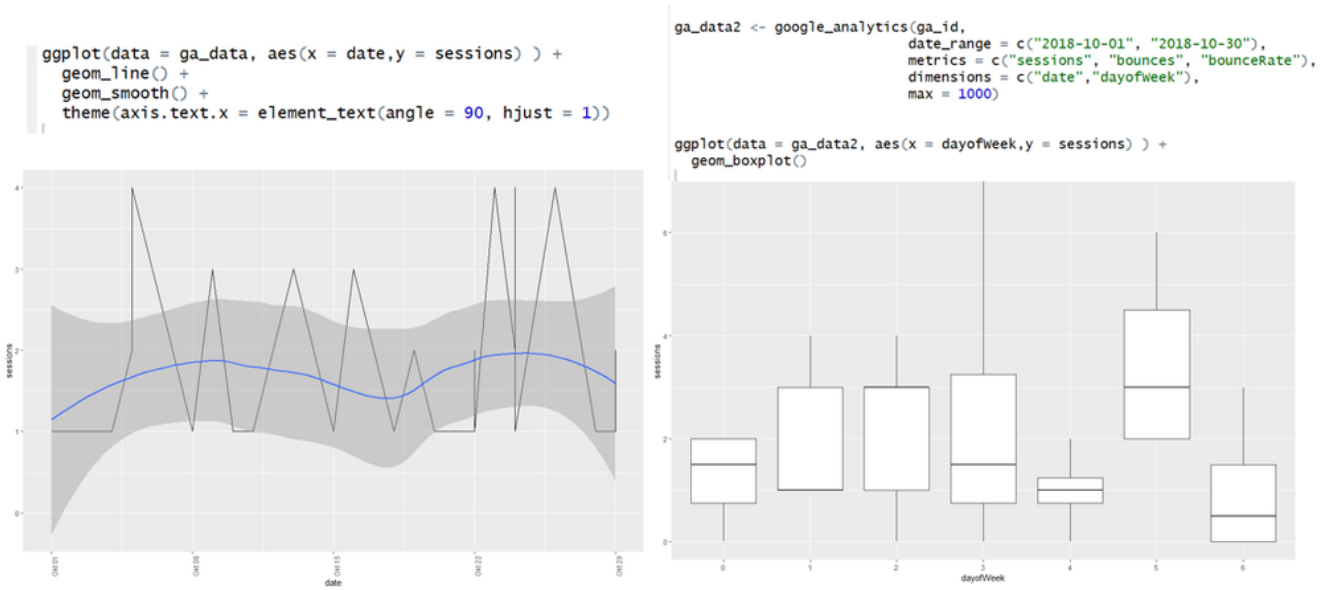


Abb. 6: Datenvisualisierungen mit dem Paket ggplot2

Tipp: Die Visualisierung bzw. Berichterstellung ist in R sehr umfassend durch sogenanntes R Markdown nutzbar. Es wird einmalig der Programmcode erstellt bzw. ein Bericht konzipiert, welcher anschließend als HTML-Seite, PDF oder Word-Dokument ausgegeben wird. Damit sind aufwendiges „Exceln“ und formatieren obsolet, die Corporate Identity ist in hohem Maße in den Berichten standardisierbar.

Richtig interessant werden Analysen und Visualisierungen, wenn diese auf Rohdaten beruhen. Mittels der Analytics API können viele Informationen verarbeitet werden, die es später zu clustern gilt. Eine sinnvolle Ergänzung ist deshalb, die Google-Analytics-Daten durch benutzerdefinierte Dimensionen zu erweitern. Wird jedem Datensatz ein exakter Zeitstempel mitgegeben, jede Session mit einer Session-ID in den Daten ergänzt sowie die Cookie-ID als Klammer für den Besucher verstanden, so könnte eine Customer-Journey-Analyse mit Rohdaten erfolgen. Simo Ahava hat in seinem Blog eine detaillierte Anleitung für diese benutzerdefinierten Dimensionen erstellt (http://einfach.st/simohava). Es ist sehr ratsam, vor der Integration den jeweiligen Datenschutzverantwortlichen zu konsultieren!

In diesem Sinne: Geben Sie R eine Chance! Im ersten Schritt hat es eine „Old-School“-Anmutung, da Befehle über eine Console eingegeben werden müssen. Mit Neugier und Training werden die Abläufe Tag für Tag vereinfacht und Webanalyse bzw. Datenvisualisierung kann wirklich auf „Knopfdruck“ erfolgen – so besteht viel mehr Freiraum und Zeit, über die Ergebnisse nachzudenken bzw. diese zu interpretieren, anstatt Zeit für die Datenaufbereitung ineffizient zu vergeuden. Und zuallerletzt: Das System wird permanent (kostenlos) weiterentwickelt– für jede Anforderung existiert bestimmt ein Paket.

Noch ein Literaturhinweis: Das kostenlose englischsprachige E-Book „Using Google Analytics with R“ von Michal Brys ist unter http://michalbrys.com/book/ in diversen Formaten abrufbar und bietet einen hervorragenden Einstieg in die Nutzung der Analytics API!

Das Powertool R – hands-on!

Das Powertool R – hands-on! –

websiteboosting.com

Da der Beitrag von Tobias Aubele in der letzten Ausgabe (Datenabzug von Google Analytics über das Tool R) auf große Resonanz gestoßen ist, möchten wir all denjenigen hier, die noch keinen Kontakt zu dem kostenlosen Tool R hatten bzw. ihn bisher gemieden haben, eine kurze und leicht verständliche…

Da der Beitrag von Tobias Aubele in der letzten Ausgabe (Datenabzug von Google Analytics über das Tool R) auf große Resonanz gestoßen ist, möchten wir all denjenigen hier, die noch keinen Kontakt zu dem kostenlosen Tool R hatten bzw. ihn bisher gemieden haben, eine kurze und leicht verständliche Einführung geben.

Keine Sorge, es wird in diesem Beitrag nicht zu technisch. Und die Beispiele sind so gehalten, dass Sie einen schnellen und effizienten Einblick in das Tool bekommen, sofern Sie nicht nur lesen, sondern sich einfach selbst mal an die Tasten setzen. Wie bei vielen mächtigen Tools ist es oft die Anfangshürde der Bedienung, an der man scheitert. Gelingt es, diese zu überwinden, eröffnet sich wie so oft ein wahres Eldorado an Vereinfachungen für die eigene Arbeit und/oder zusätzliche Möglichkeiten für nötige Analysen oder die Gewinnung wichtiger Erkenntnisse.

Eigentlich ist R als Statistiktool konzipiert und bekannt. Aber im Lauf der Jahre ist durch eine schier unendliche Anzahl an Funktionsbibliotheken die Anwendungsbreite förmlich explodiert. So kann man z. B. Ergebnisse automatisch in HTML-Webseiten übertragen oder Textmining betreiben. Zwei zentrale Stärken von R sind wohl zum einen die Anzahl der Daten, die man damit sehr schnell verarbeiten kann. Zum anderen kann man aber alles, was man für ein Projekt oder eine Analyse experimentell durchgeführt hat, auf Knopfdruck ganz oder teilweise mit anderen Daten wiederholen. Richtig eingesetzt, kann das gerade im Online-Marketing sehr viel Zeit und damit Kosten sparen. Natürlich sind Sie nach der Lektüre des Beitrags kein R-Spezialist – aber dann können Sie für sich entscheiden, ob Sie dem Tool eine Chance geben und sich etwas

tiefer darin eingraben. Im Web gibt es viele Tutorials, bei Udemy einen deutschen Kurs dazu und am Ende dieses Beitrags finden Sie Literatur für den Einstieg und für Fortgeschrittene zum Thema Data Science.
Erwarten Sie persönlich, dass die Menge der Daten künftig an Ihrem Arbeitsplatz eher zunehmen wird? Wird die Zahl der Datenquellen eher größer und haben diese wahrscheinlich unterschiedliche Strukturen? Schreitet die Digitalisierung bei Ihnen gut voran? Dann, ja dann kann es sich wirklich lohnen, sich frühzeitig mit so einem Tool auseinanderzusetzen, das Sie bei der Bewältigung gut unterstützt. Und wie immer gilt: Wer früher dran ist, kann den anderen die lange Nase zeigen, während die sich wundern, wie man das Kaninchen mal wieder aus dem (neuen) Hut gezaubert hat …

Was ist R überhaupt?

R selbst ist eigentlich eher eine Programmiersprache als ein Tool und ähnelt Python. Über das ebenfalls kostenlose RStudio erhält R eine komfortable und übersichtliche Benutzeroberfläche, die Einsteigern etwas den Schrecken nimmt. Ursprünglich für statistische Anwendungen gedacht, ist R mittlerweile sehr viel mehr und hat große Stärken, wenn man Daten auswerten, umwandeln oder visualisieren muss. Sie möchten wissen, wie oft das Wort „Westernstiefel“ auf einer bestimmten Webseite vorkommt? Kein Problem – mit den entsprechenden kostenlosen Erweiterungspaketen (Librarys) ist das mit einer Befehlszeile erledigt. Sie brauchen einen Überblick über die Struktur der Shop-Umsatzdaten des letzten Jahres? Dazu laden Sie die Umsatzdatei ein und der Befehl „sum“ gibt Ihnen im Bruchteil einer Sekunde statistisch relevante Informationen. Vorausgesetzt, Sie wissen, was ein Median oder eine Standardabweichung ist. Mit anderen Worten sehen Sie mit einer Zeile Code, ob z. B. der häufig falsch verwendete Mittelwert als Kennzahl bei Ihren Daten überhaupt aussagekräftig ist.

„Windows, Mac oder Linux? Geht alles!“

Wer tiefer einsteigt, kann mit R auch Anwendungen für Machine Learning fahren oder Data Mining betreiben, indem man im einfachsten Fall über die Daten Regressionsanalysen laufen lässt. Bereits mit zwei bis drei Befehlszeilen kann man sich durch Umwandlung von Daten einen visuellen Überblick über mögliche Abhängigkeiten verschaffen. Ist der Umsatz tatsächlich vom Wetter abhängig und wenn ja, wie stark? Wie wirken sich Preisveränderungen von Mitbewerbern auf den eigenen Absatz aus und wie eine Erhöhung des Ads-Budgets? Gibt es Abhängigkeiten, die man bisher noch gar nicht auf dem Schirm hatte und wo es sich lohnt, tiefer zu graben?

R und RStudio sind für Windows, Mac OS und Linux verfügbar.

Was ist das Besondere an R?

Wie schon erwähnt, machen die vielen Funktionsbibliotheken (Librarys) R erst richtig nützlich und breit anwendbar. Diese lassen sich sehr simpel bei Bedarf einbinden und stehen fortan im Tool zur Verfügung. Damit kann man dann z. B. bestimmte Dateiformate lesen und erzeugen, Daten direkt in HTML-Vorlagen ausgeben, Anbindungen an verschiedene Datenbanken managen, linguistische Analysen durchführen oder auch Marktforschung betreiben. Und das Ganze völlig kostenlos, sehr stabil und performant.

Eine Besonderheit gegenüber anderen Programmiersprachen liegt darin, dass man in R mit sog. Vektoren arbeitet. Das erleichtert den Umgang mit Daten und es sind nicht wie sonst gesondert programmierte Schleifen nötig. Einen Vektor kann man sich als komplette Zeile (oder Spalte) in Excel vorstellen, der sich direkt verarbeiten lässt. Im Prinzip lässt sich so z. B. der komplette Quellcode einer Webseite in einen Vektor einlesen und über den Namen des Vektors ansprechen.

„R ist kostenlos, stabil und performant.“

Wie Sie später noch sehen werden, ist es recht einfach, alle Befehle, die man in die Console eingegeben hat, später wieder aufzurufen und ggf. bei Bedarf in Summe in einen individuellen Programmcode zu überführen. Das erleichtert das Verständnis, die Erstellung eines kleinen Programms und auch die Fehlersuche. Hier greift das gleiche Grundprinzip wie beim Arbeiten mit Excel, wenn man kein ausgebildeter bzw. fachkundiger Programmierer ist. Man bearbeitet Daten Schritt für Schritt und sieht jeweils sofort das Ergebnis und ob alles wie gewünscht passt. Nach Eingabe einer fehlerhaften Formel erscheint sofort eine Fehlermeldung – nicht erst, wenn später ein Programm läuft. Erst wenn alles passt, speichert man sich die einzelnen Befehlszeilen ab und lässt später direkt alle auf einmal ablaufen.

R und RStudio enthalten eine umfassende Hilfe: Durch Eingabe eines Fragezeichens, gefolgt von einem Funktionsbefehl wird die Hilfe aufgerufen, die die Verwendung genau erklärt. Auch im Reiter „Help“ (Abbildung 2; Ziffer 8) ist ein kleines Suchfeld integriert, das die Suche erleichtert. Empfehlenswert für einen strukturierten Einstieg ist es allerdings, in ein entsprechendes Fachbuch zu investieren. Eine kleine Auswahl finden Sie am Ende dieses Beitrags.

„R ist sehr flexibel bei der Datenausgabe.“

R hat sehr flexible Ausgabeformate. So können Ergebnisse, Tabellen, Grafiken und anderes über Erweiterungspakete z. B. als CSV, TXT, XLS, Word, PowerPoint oder auch direkt in HTML ausgegeben werden (Abbildung 1). Ebenso ist es natürlich möglich, die Daten in angebundene Datenbanken zu schreiben. Damit entfällt der sonst übliche Anwendungsbruch, weil man z. B. in Excel arbeitet, Diagramme erzeugt und diese dann für eine Präsentation in PowerPoint kopiert. Wer seine Berichte modern lieber gleich per HTML ins Intranet oder ins Web

schießen möchte, kann das ebenfalls tun. Analyse, Aufbereitung und Dokumentation können also ein einem einzigen Tool erledigt und alle nötigen Schritte dazu reversibel zentral mit abgespeichert werden.

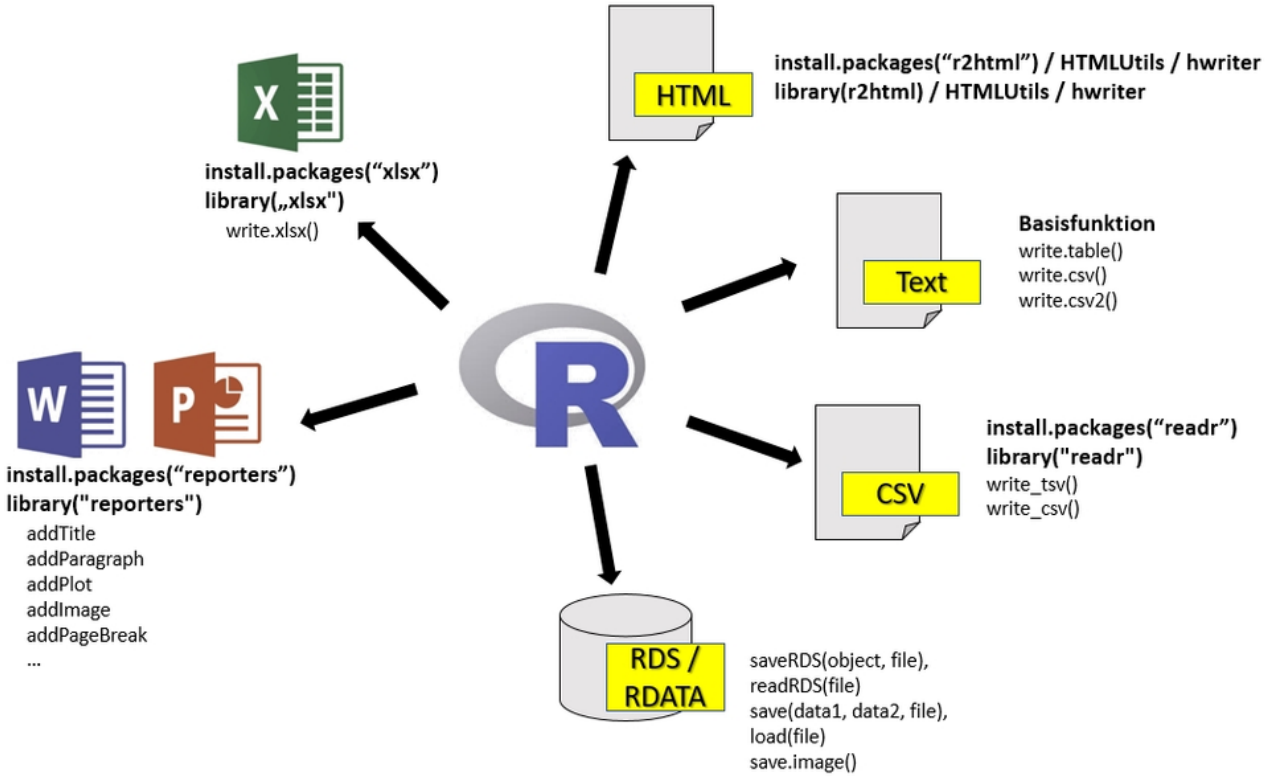


Abbildung 1: Auszug für gängige Ausgabe- bzw. Exportformate in R

„Nichts geht mehr verloren – kein einziger Arbeitsschritt."

Vielleicht ist einer der größten Vorteile von R, dass Sie alles zusammen in einer Umgebung halten. Wie laufen üblicherweise Analysen im weitesten Sinne in Unternehmen ab? Man hat ein Set von Daten in verschiedenen Formaten und nicht selten strukturell unterschiedlich. Man kopiert bzw. liest diese dann oft in Excel ein, wandelt bestimmte Daten um, verdichtet, sortiert, filtert und modifiziert sie, führt weitere Konsolidierungen und Berechnungen durch und erzeugt diverse Diagramme oder Übersichtstabellen. Werden diese in einen Bericht gegossen oder präsentiert, kopiert man sie in Word oder PowerPoint. So weit, so gut. Im nächsten Monat/Quartal/Jahr oder beim nächsten Projekt steht dann eine Wiederholung des Ganzen an. Dabei entsteht oft das Problem,

dass man nicht nur all diese Arbeiten erneut durchführen muss. Noch öfter steht man vor den fertig modifizieren Tabellen und weiß nicht mehr, wie man von den Ursprungsdaten dort hingelangt ist, da nicht mehr alle Formeln vorhanden sind bzw. neu erdacht und ausprobiert werden müssen. Alles in allem eine gigantische und auch ärgerliche Verschwendung von Zeit und Ressourcen.

Wenn Sie sich darauf einlassen, R etwas näherzukommen, gehört dies künftig der Vergangenheit an. Denn in R speichern Sie bei Bedarf jeden einzelnen Schritt – das Öffnen der Datei(en), jede Bearbeitung und schließlich auch die Erzeugung von Ergebnissen, Kennzahlen, Diagrammen und visuellen Auswertungen. Alles wird in einer Umgebung abgelegt und kann jederzeit nachvollzogen und reproduziert werden. Wenn Sie tiefer in R eintauchen, werden Sie sogar in der Lage sein, jedwede Auswertung automatisch per HTML oder PowerPoint neu zu erzeugen. Im einfachsten Fall kopieren Sie eine immer gleich benannte Datei mit den aktuellen Monatszahlen in das Arbeitsverzeichnis von R und starten Ihre aufgezeichnete Programmierung. Das war es dann auch schon. Ein kompletter und durchaus auch optisch anspruchsvoller Bericht steht Sekunden später vollautomatisch im Intranet.

Wo finde ich R und RStudio?

Die Installation von R ist recht einfach. Es lässt sich kostenlos auf der Website von

cran.r-project.org

herunterladen, und zwar als Windows-, Mac-OS- und Linuxversion. R selbst ist wie erwähnt nicht so komfortabel zu bedienen und daher empfiehlt es sich für die meisten Nutzer, gleich noch RStudio dazu zu installieren. RStudio muss *nach* R installiert werden, sucht dann aber automatisch nach der R-Installation und integriert das Tool in eine übersichtlichere Benutzeroberfläche mit vielen nützlichen zusätzlichen

Funktionen. Die Basisversion ist ebenfalls frei und unter rstudio.com erhältlich:

einfach.st/rstudio.

Wer den entsprechenden Rechner hat, sollte sich gleich die 64-Bit-Version installieren bzw. verwenden, da gerade rechenintensive Analysen deutlich schneller sind und auch die üblichen Restriktionen beim Datenvolumen unter 32 Bit entfallen.

Für normale Anwendungen ist RStudio kostenlos. Es gibt allerdings auch kommerzielle Lizenzen mit Support und weiteren Features sowie eine Serverversion.

Beim ersten Start prüft RStudio bzw. R noch, ob auf den Rechner eine aktuelle Java-Version installiert ist, was aus Sicherheitsgründen sowieso auf jedem Rechner Pflicht sein sollte. Anschließend präsentiert sich R integriert in RStudio aufnahmebereit für die ersten Befehle bzw. Gehversuche.

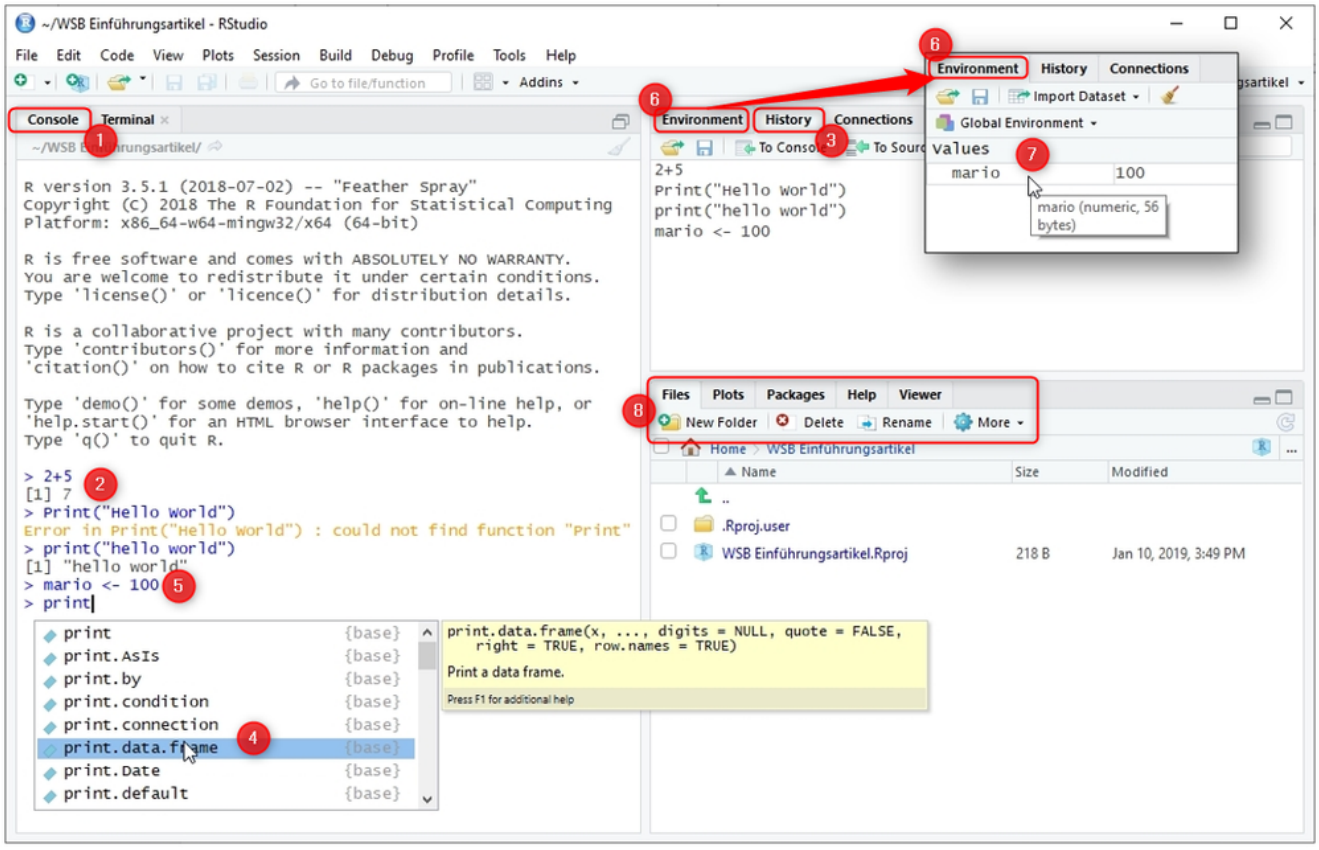


Abbildung 2: Der Startbildschirm von RStudio

Legen Sie jetzt einfach los

Nach der erfolgreichen Installation kann man sofort mit R bzw. RStudio arbeiten (Abbildung 2). Im linken Teil befindet sich die Console (Ziffer 1), in der die Befehle eingetippt werden. Geben Sie nach dem Promt (der blinkende Cursor nach dem „>“ Zeichen) einfach

```
2+5
```

ein und drücken Sie Return (Ziffer 2). Als Antwort erhalten Sie das Ergebnis dieser einfachen Rechenoperation. Gibt man einen Befehl falsch ein, z. B. „Print“ mit einem Großbuchstaben, erscheint eine Fehlermeldung, die hier zur Demonstration erzwungen wurde. Normalerweise erscheint nämlich bereits beim Tippen ein Vorschlagfenster mit den gültigen Befehlen für Funktionen (Ziffer 4). Das bedeutet, man kann sich den Rest einer Anweisung wie z. B. „print.data.frame“ sparen und den gewünschten Befehl einfach übernehmen.

Tipp

Wenn der Cursor im Consolenfenster blinkt, können Sie mit den Cursortasten hoch und runter die letzten eingetippten Befehle holen bzw. in diesen blättern. Das hilft insbesondere bei Tippfehlern, damit nicht alles erneut eingegeben werden muss. Einfach mit der Cursortaste-oben den letzten (falschen) Befehl holen, an der entsprechenden Stelle ausbessern und mit Return erneut auslösen. Zudem zeichnet R jeden Befehl in einer Historie auf, sodass er jederzeit auch später einfach noch mal verwendet werden kann.

Variablen weist man mit „<-“ ganz einfach einen Wert zu. Das machen wir hier nur einmal beispielhaft, um die Vorzüge des RStudios zu zeigen. Ziffer 5 zeigt eine solche Zuweisung:

```
wsb <- 100
```

weist der Variable „wsb“ damit den Wert 100 zu. Also erst der gewünschte Name, dann die Zuweisungszeichen und anschließend der Wert. Im rechten oberen Teil von RStudio findet man dann unter dem Reiter „Environment“ (Ziffer 6 und das angezeigte Fenster mit Ziffer 6) genau diese und natürlich alle anderen Variablen wieder, die man im Lauf einer Sitzung definiert hat. Das dient der Kontrolle, welchen Wert eine Variable gerade hat. Wenn Sie jetzt als Befehlszeile

```
wsb + 5
```

gefolgt von Return eingeben, sehen Sie unter Environment rechts oben den neuen Wert 105 für diese Variable. Prinzipiell kann man ganzen Datenpools (Vektoren, Matrizen, Data Frames) Variablennamen zuweisen, was das Handling und die Lesbarkeit durch einfache Namensvergabe enorm vereinfacht. Dies aber nur als Hinweis.

Unter dem Reiter „History“ (Ziffer 3) werden alle Befehle aufgezeichnet, die man in die Console eingegeben hat. Mit einem Mausklick lassen sie sich einfach wiederholen bzw. in die Console übernehmen. Später lassen sich diese Befehle dann ganz einfach in einem kleinen Programmcode überführen, der nacheinander abläuft.

Das charmante Prinzip ist also für den Einsteiger, dass er zunächst Schritt für Schritt nach jeder Eingabe explorativ prüfen kann, ob alles so richtig ist, und erst danach fasst man quasi die einzelnen Befehle zu einem ablauffähigen Programm zusammen und kann dieses dann mit einem einzigen Befehl starten.

Einfach mal machen!

Alle hier gezeigten Beispiele sind so gehalten, dass Sie möglichst einfach selbst erste Schritte zum Ausprobieren unternehmen können. Wir haben daher oft auf branchenübliche Zahlenbeispiele verzichtet, weil R einige Datenquellen zum

Experimentieren bereits mitliefert. Sie müssen sich also nicht erst mit dem Erstellen und Einlesen von Datensätzen herumschlagen, nur um überhaupt eine Grundlage zum Testen zu haben. Ob Sie nun die Kelchlänge von Lilienarten oder den Benzinverbrauch von Autos visualisieren oder Marketingausgaben oder Backlinks, ist von der Methode her ja austauschbar. Hier liegt der Fokus tatsächlich darin, Sie sofort in die Lage zu versetzen, einfach einmal mit dem Tool herumzuspielen.

Ein Anwendungsbeispiel: Sie nutzen regelmäßig einen oder mehrere Datensätze, z. B. aus einem SEO-Tool, mit denen Sie arbeiten müssen. Diese enthalten aber unerwünschte (Sonder-)Zeichen, wie Umlaute als &szlig“ statt einem „ß“ oder schlicht den typischen Punkt statt des deutschen Kommas als Kommastelle. Das kann man natürlich auch in Excel erledigen, aber man muss alle Schritte einzeln und jedes Mal durchführen. Über R lädt man sich so einen Datensatz, verändert ihn mit den entsprechenden Anweisungen einmalig und prüft, ob alles korrekt ist. Anschließend speichert man sich die Befehle einfach ab und kann sie später direkt aufrufen. Der Datensatz muss dann nur noch geladen werden, das entsprechend passende erstellte Programm wird gestartet und der Datensatz entweder manuell oder sogar schon vom eigenen Programm abgespeichert – fertig. Prinzipiell lassen sich damit alle Daten auf einfache Weise verändern, bereinigen, auf Konsistenz prüfen oder einfach nur einzelne Zeichen tauschen.

Im rechten unteren Bereich von RStudio (Ziffer 8) hat man Zugriff auf Datenfiles, Grafiken (Plots), Packages (Erweiterungen) sowie auf eine Hilfe und einen Viewer.

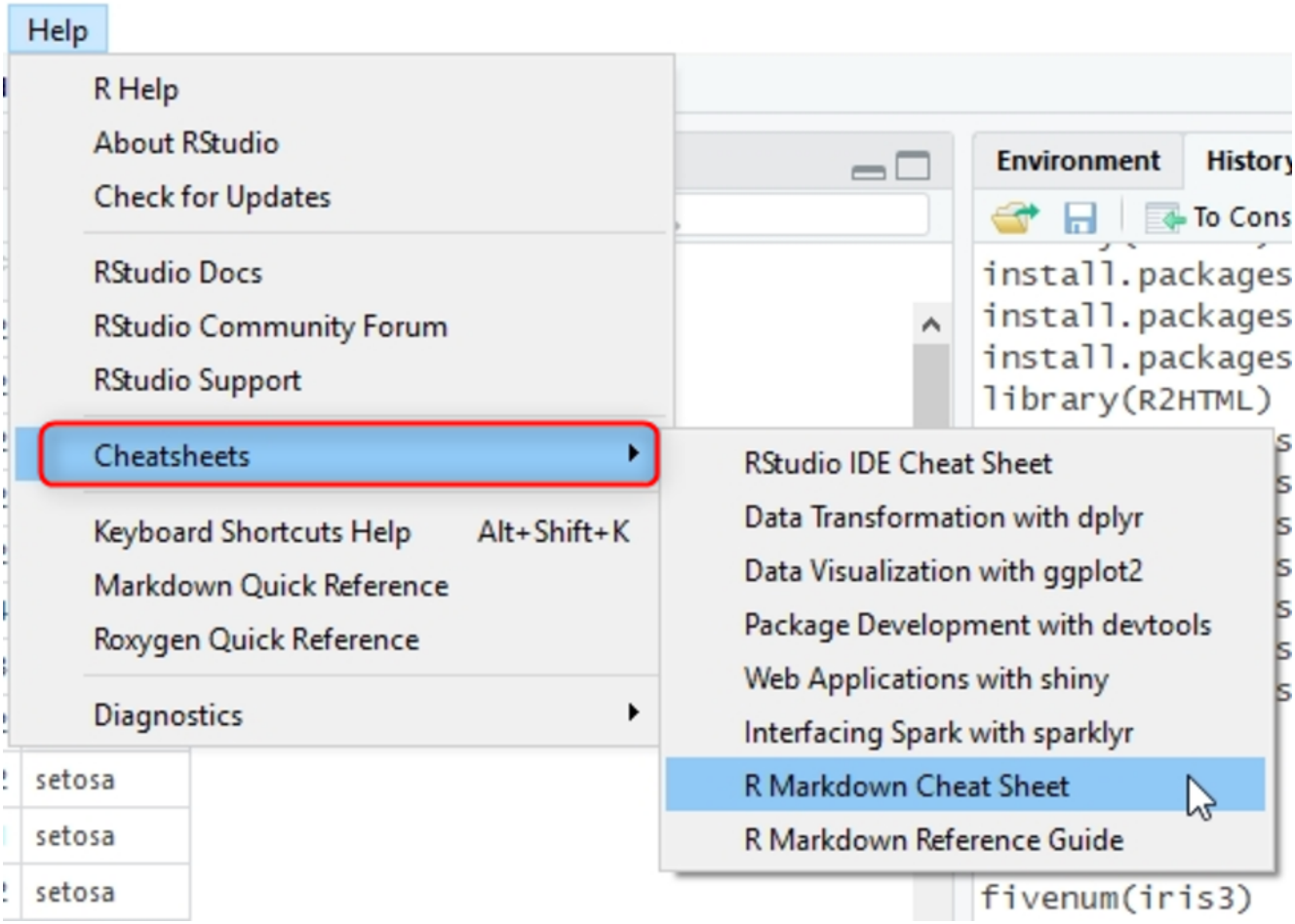


Abbildung 3: In RStudio findet man unter dem Menüpunkt „Help“ nützliche Cheatsheets

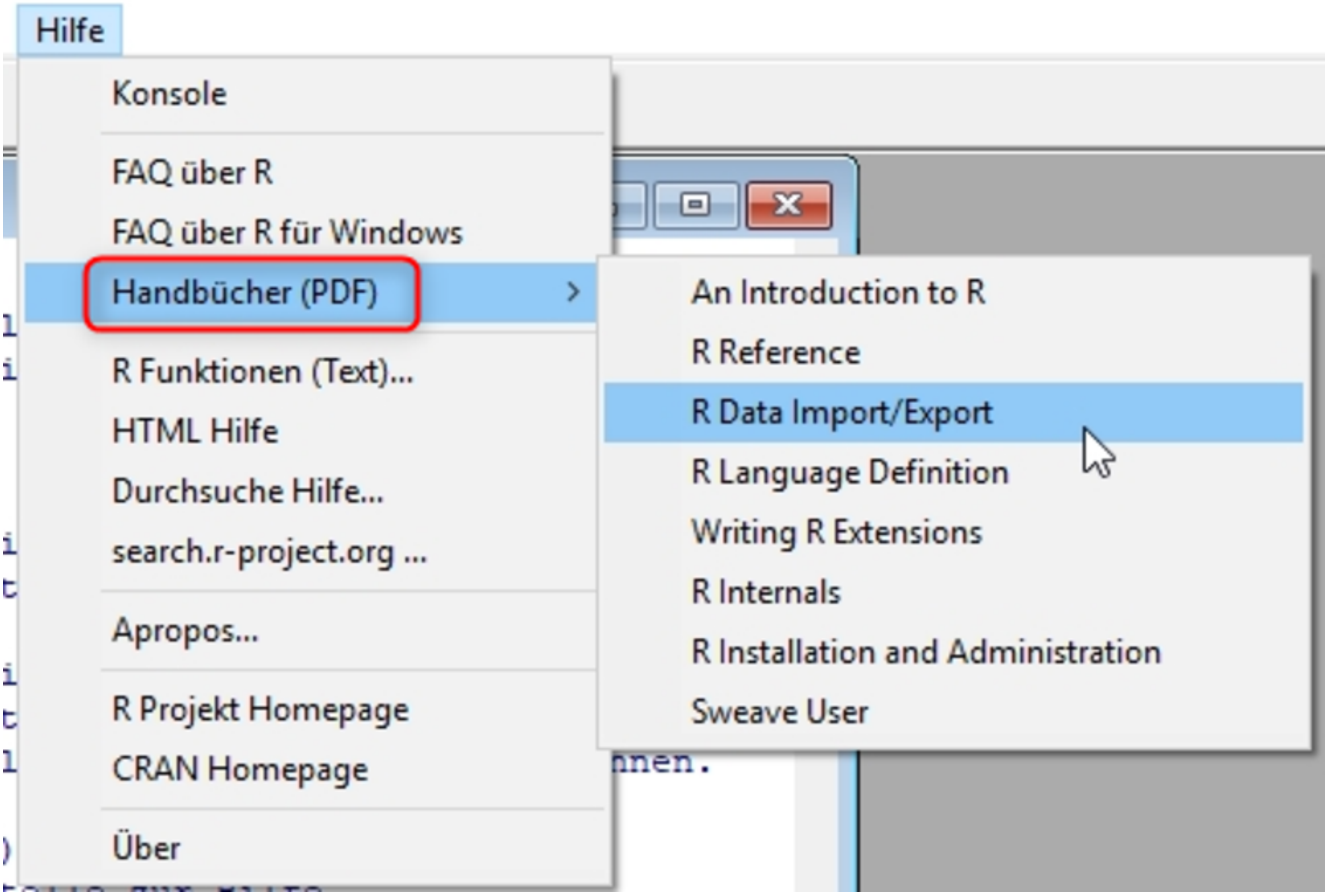


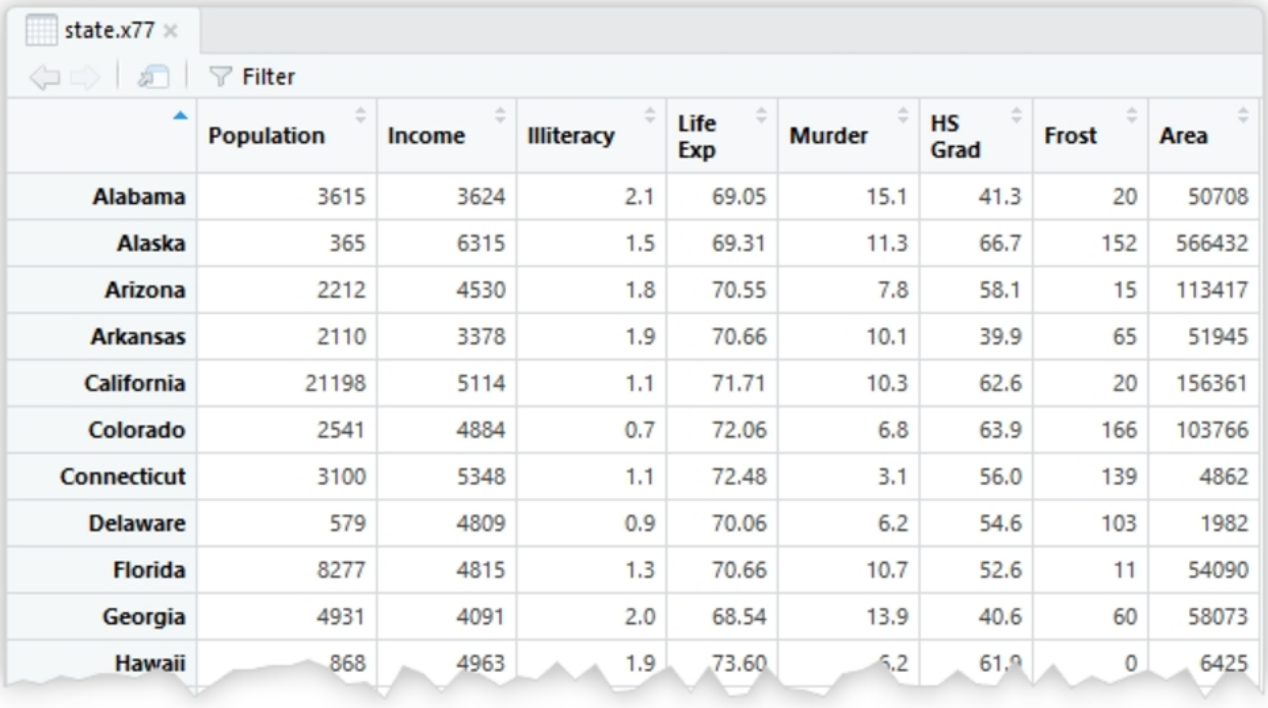
Abbildung 4: In R selbst (nicht in RStudio) sind umfangreiche PDF-Handbücher hinterlegt (englisch)

Mitgelieferte Testdateien

Praktischerweise beinhaltet die Installation von R bereits einige Dateien mit Daten, sodass man die ersten Gehversuche unternehmen kann, ohne erst einmal selbst Daten einlesen zu müssen. Eine dieser Beispieldatensätze ist „state.x77“. Er lässt sich mit dem Befehl

```
View(state.x77)
```

anzeigen. Links oben im Fenster werden jetzt alle 50 US-Staaten mit einigen Spalten wie „Einwohnerzahl“, „Durchschnittseinkommen“ etc. angezeigt (Abbildung 5).


state.x77

Filter

	Population	Income	Illiteracy	Life Exp	Murder	HS Grad	Frost	Area
Alabama	3615	3624	2.1	69.05	15.1	41.3	20	50708
Alaska	365	6315	1.5	69.31	11.3	66.7	152	566432
Arizona	2212	4530	1.8	70.55	7.8	58.1	15	113417
Arkansas	2110	3378	1.9	70.66	10.1	39.9	65	51945
California	21198	5114	1.1	71.71	10.3	62.6	20	156361
Colorado	2541	4884	0.7	72.06	6.8	63.9	166	103766
Connecticut	3100	5348	1.1	72.48	3.1	56.0	139	4862
Delaware	579	4809	0.9	70.06	6.2	54.6	103	1982
Florida	8277	4815	1.3	70.66	10.7	52.6	11	54090
Georgia	4931	4091	2.0	68.54	13.9	40.6	60	58073
Hawaii	868	4963	1.9	73.60	6.2	61.9	0	6425

Abbildung 5: R liefert einige Beispieldaten automatisch mit
Ein einfacher Befehl zeigt Ihnen direkt statistisch relevante Daten dieses Datensatzes an (Abbildung 6):

```
summary(state.x77)
```

R zeigt für jede Spalte gesondert die statistisch wichtigsten

Strukturinformationen an. Für eine spätere Datenanalyse oder die Bildung von Kennzahlen ist dies sehr nützlich, weil man sofort u. a. die Streuung ablesen kann. Ein extremes Beispiel zur Verdeutlichung: Hatten 200 Verkäufe den Wert 1 € und weitere 200 den Wert 100 €, dann macht die Verwendung eines Mittelwerts (50 €) nicht wirklich Sinn.

```
> summary(state.x77)
   Population        Income       Illiteracy      Life Exp        Murder          HS Grad
 Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96   Min.   : 1.400   Min.   :37.80
 1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12   1st Qu.: 4.350   1st Qu.:48.05
 Median : 2838   Median :4519   Median :0.950   Median :70.67   Median : 6.850   Median :53.25
 Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88   Mean   : 7.378   Mean   :53.11
 3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89   3rd Qu.:10.675   3rd Qu.:59.15
 Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60   Max.   :15.100   Max.   :67.30
     Frost             Area
 Min.   :  0.00   Min.   :  1049
 1st Qu.: 66.25   1st Qu.: 36985
 Median :114.50   Median : 54277
 Mean   :104.46   Mean   : 70736
 3rd Qu.:139.75   3rd Qu.: 81163
 Max.   :188.00   Max.   :566432
```

Abbildung 6: Statistisch relevante Strukturdaten der Datei „state.x77″

Der Befehl

```
state.x77[11,2]
```

greift z. B. direkt im Datensatz „state.x77“ auf die elfte Zeile und die zweite Spalte zu und liefert 4963 als das Durchschnittseinkommen in Hawaii zurück (in Abbildung 5 zu sehen). Genauso können über Spalten oder Zeilen natürlich entsprechende weitere einfache Berechnungen oder Analysen durchgeführt werden.

Die ebenfalls in R mitgelieferte Datei „state.name“ beinhaltet nur die Namen der 50 US-Staaten. Welche dieser Namen besteht aus mehreren Wörtern wie z. B. New Jersey? Hier hilft die Funktion „grep“, mit der man in diesen Daten ganz einfach nach einem Leerzeichen sucht. Beachten Sie bitte, dass sich der Name der verwendeten Datendatei jetzt ändert bzw. eine andere Datei (state.name) zugrunde liegt (Testen Sie doch mal View(state.name) um zu sehen, was der Datensatz beinhaltet.)

```
state.name[grep(“ „, state.name)]
```

Als Ergebnis erhält man in der Ausgabe die zehn Staaten mit einem „Doppelnamen“. Tauschen Sie das Leerzeichen z. B. gegen den Wortbestandteil „New“ aus, erhalten Sie die vier Bundesstaaten, die eben diese Zeichenfolge enthalten, der Befehl lautet dann:

```
state.name[grep(„New“, state.name)]
```

Diese einfachen Beispiele sollen nur verdeutlichen, wie leicht die Extraktion und Filterung in Datenbeständen ist und welches Prinzip dahintersteckt. Natürlich kann man solche Aufgaben bei so kleinen Datensätzen auch gut und einfach z. B. in Excel erledigen. Werden die Daten jedoch umfangreicher, macht das schon deutlich mehr Arbeit, die zudem prinzipiell jedes Mal neu durchgeführt werden muss. Die Befehle bzw. Befehlssequenzen hier in R können jedoch abgespeichert und zu jedem späteren Zeitpunkt einzeln oder in Summe aufeinanderfolgend gestartet werden.

Tipp: Geben Sie den Befehl

```
datasets::
```

ein und Sie erhalten in einem Pop-up-Fenster die in R integrierten Versuchsdatensätze angezeigt, die Sie per Mausklick einfach übernehmen können (Abbildung 7). Geben Sie den Befehl

```
data()
```

ein, erscheinen im linken oberen Fenster alle Namen und eine jeweils kurze Beschreibung aller Datensets, die bereits installiert sind.

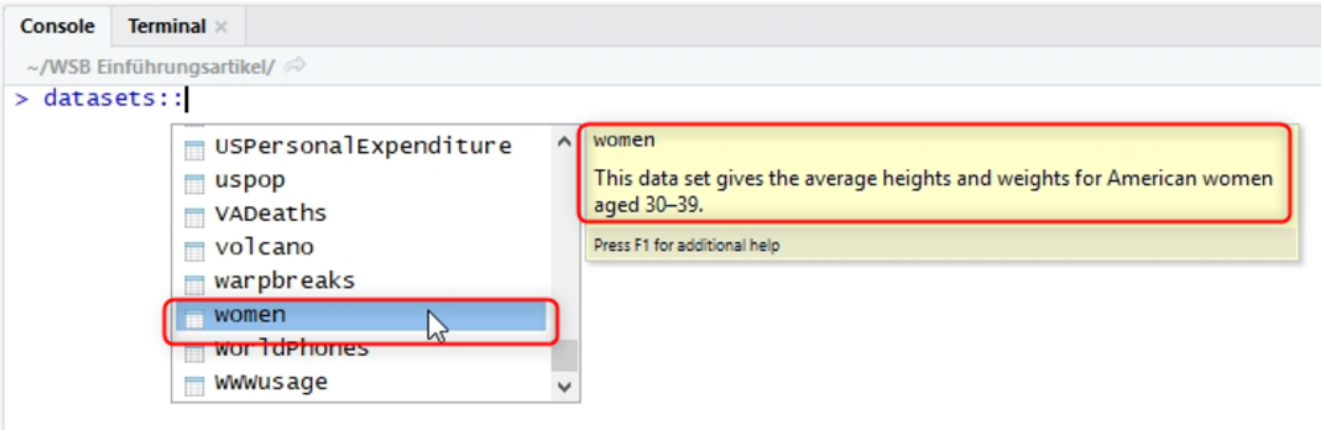


Abbildung 7: Automatisch mitgelieferte Datensätze zum Ausprobieren mit Inhaltsbeschreibung

Noch ein weiteres kleines Beispiel zeigt Abbildung 8. Für die Variable „Zeichenkette“ wird zunächst hilfsweise ein Text „Die Website Boosting …“ eingelesen. Der Befehl „strsplit“ zerlegt dann den Text in einzelne Wörter bzw. trennt nach einem Leerzeichen. Stellen Sie sich vor, Sie hätten eine Exceltabelle mit mehreren Tausend Zellen, die jeweils den Text einer Webseite enthalten, und für semantische Analysen müsste jedes Wort extrahiert werden. In R liest man dazu vereinfacht erklärt über ein Erweiterungspaket ein Set von URLs mit einer einzigen Funktion in Vektoren ein und ein weiterer Befehl extrahiert aus jedem Vektor jedes einzelne Wort. Bereits mit einigen Grundkenntnissen in R lässt sich so durchaus enorm Zeit sparen.

Unter einfach.st/rdatasets finden Sie übrigens eine recht gut kommentierte Übersicht (englisch) mit Beispielen und Beispielcode für die mitgelieferten Data Sets.

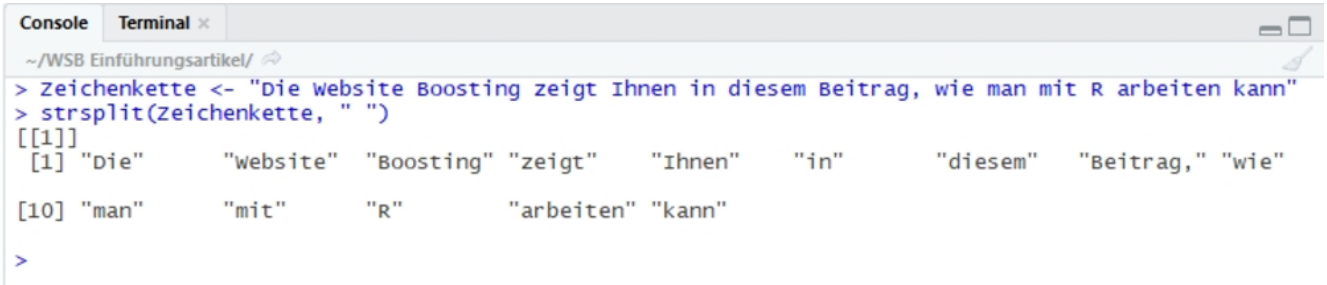


Abbildung 8: Mit einem einzigen Befehl Texte in Wörter zerlegen

Erweiterungspakete installieren

Wie erwähnt werden für R sehr viele Erweiterungspakete im Web angeboten, die sich recht einfach integrieren lassen. Diese muss man nur einmal zu R hinzuladen und kann sie später dann nach dem jeweiligen Programmstart bei Bedarf aktiveren und die dort hinterlegten Funktionen nutzen. Eine erste gute Quelle finden Sie auf cran.r-project.org. Die dort ladbaren Pakete kann man direkt über R laden. Dafür muss bzw. sollte man natürlich den Namen der Erweiterung kennen. Diese werden meist in Anleitungen, Beispielen oder in der Literatur erwähnt. Die Installation kann dann per Befehlszeile über die Funktion „install.packages“ erfolgen (hier die Erweiterung „htmltidy“ zum Umgang mit HTML-Dokumenten):

install.packages(„htmltidy“)

oder über das Menü oben in RStudio unter Tools/Install Packages, wie in Abbildung 9 zu sehen ist. Auch hier erscheint beim Tippen bereits ein kleines Pop-up mit einer entsprechenden Auswahl. Lässt man den Haken „Install depencies“ (in der Abb. verdeckt unter dem Pop-up) drin, werden sog. abhängige Packages gleich mit installiert. Die Erweiterung „htmltidy“ greift z. B. unter anderem auf die Erweiterung „xml“ und „htmlwidgets“ zu und löst damit auch gleich deren automatische Integration aus. In der Console erscheint dann eine entsprechende Meldung.

Tipp

Manchmal zickt R bei der Eingabe von Erweiterungspaketen über die Kommandozeile mit einer Fehlermeldung zurück, weil die Groß-/Kleinschreibung nicht passt. Daher empfiehlt es sich tatsächlich, das (einmalige) Laden von Packages – aus Sicht eines Programmierers sicher unehrenhaft, aber eben einfacher – über das Menü von RStudio wie in Abbildung 9 gezeigt zu erledigen.

Die meisten Packages sind recht gut dokumentiert. So findet man für das eben erwähnte Package weitere Infos unter cran.r-

project.org/web/packages/htmltidy/index.html und dort unter „Downloads“ auch ein PDF mit einer genauen Funktionsbeschreibung.

Möchte man die Funktionen eines einmal installierten Packages in R verwenden, aktiviert man es ganz einfach mit der Anweisung:

library(packagename) – im Beispiel also: library(htmltidy)

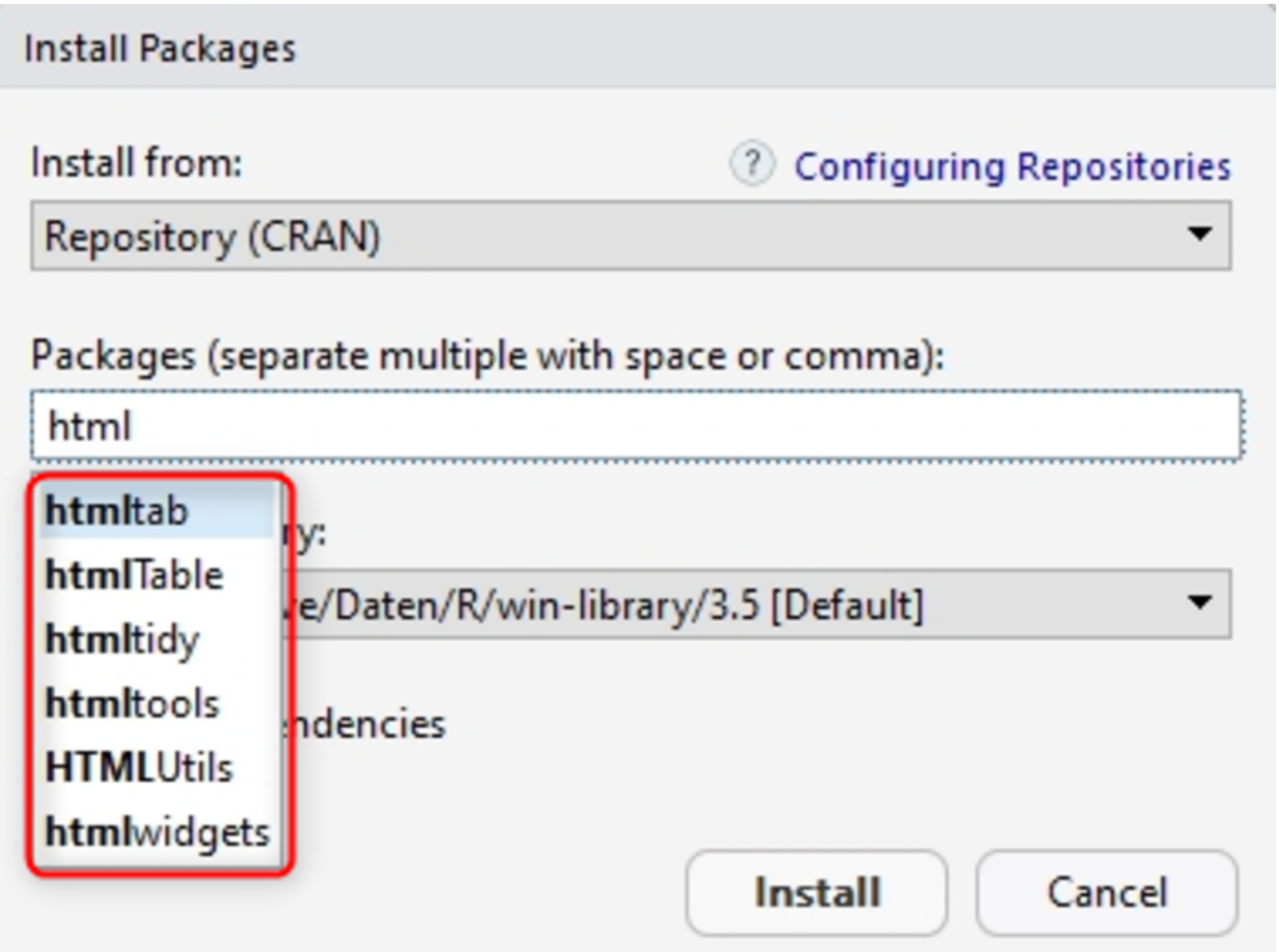


Abbildung 9: Erweiterungen lassen sich sehr einfach installieren

Einen Überblick über installierte Erweiterungen findet man in RStudio im rechten unteren Fenster unter dem Reiter „Packages“ (siehe Abbildung 2, Ziffer 8).

Eine CSV-Datei einlesen,

modifizieren und wieder ausgeben

Zum Einlesen von Dateien ist es nützlich, gleich zwei Erweiterungen zu installieren: „readr“ und „readxl“ (für das Excelformat). Dazu geben Sie einfach die beiden Befehlszeilen ein

install.packages(„readr“)

install.packages(„readxl“)

und aktivieren anschließend zumindest die Erweiterung „readr“:

library(readr)

Erstellen Sie zum Ausprobieren eine einfache CSV-Datei z. B. über Excel oder einen Texteditor mit mehreren Zeilen und Spalten. In die erste Zeile schreiben Sie durch Kommata getrennt die Spaltenüberschriften und darunter ebenfalls durch Kommata getrennt die Datensätze, wie z. B.

URL,Statuscode,Inlinks
www.meine-domain.de,200,1.250
www.deine-domain.com,200,3.243
www.ihre-domain.at,200,13.283

Der Punkt bei den Inlinks-Zahlenwerten ist absichtlich gewählt, er soll durch ein Komma getauscht werden, um das Prinzip der Datenmodifikation zu zeigen. Erzeugen Sie in RStudio über „File“ – „New Projekt“ ein neues Projekt und speichern Sie dieses auf dem Rechner unter einem beliebigen Ort ab. Der angegebene Projektname ist dann automatisch der Verzeichnisname und Ihr „Working Directory“. In dieses Verzeichnis speichern Sie dann die CSV-Datei ab, damit Sie diese durch einen einfachen Befehl öffnen und später speichern können. Alternativ kann man mit R natürlich auch auf alle anderen Verzeichnisse/Dateiorte zugreifen, aber die Dateien zu Anfang in einem eigenen Working Directory zu speichern, erleichtert die Arbeit etwas. In dem Beispiel hier wurde als

Working Directory „WSB Einführungsartikel“ gesetzt und rechts unten unter „Files“ taucht die Datei beispiel.csv nach der Erzeugung bzw. dem Abspeichern dann auch entsprechend auf (Abbildung 10).

Der Befehl

getwd()

gibt übrigens das aktuelle Verzeichnis aus.

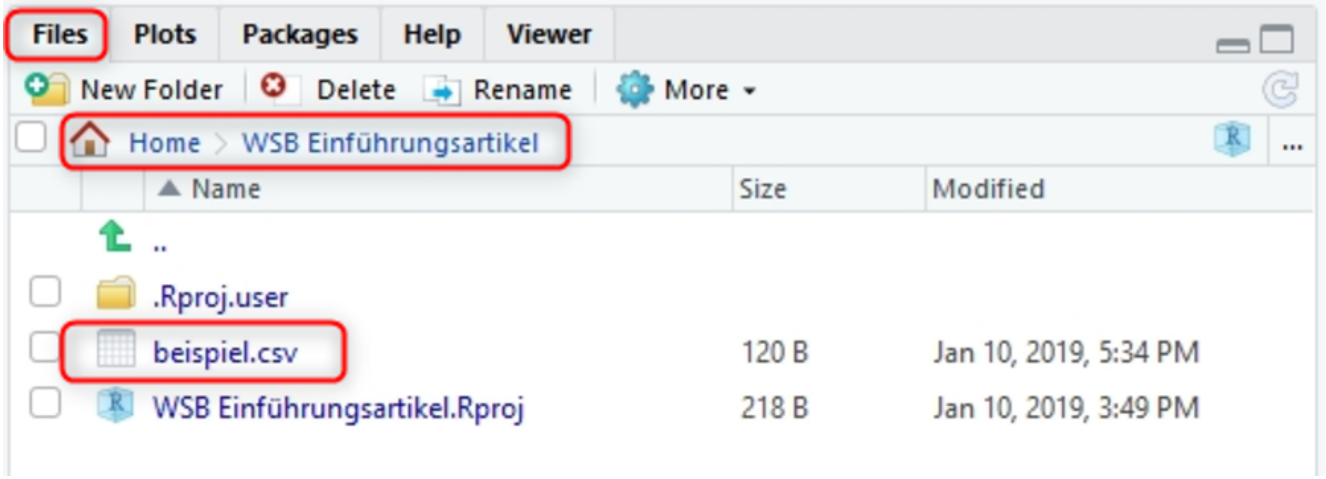


Abbildung 10: Legen Sie über ein neues Projekt einen Speicherort zum Testen an
Über den Befehl

beispiel <- read.csv(„beispiel.csv“)

weisen Sie jetzt ganz einfach der Variable „beispiel“ (oder auch einem beliebigen anderen Namen) über die Funktion „read.csv“ die Datei beispiel.csv zu. Da die Datei im Working Directory liegt, brauchen Sie keine gesonderten Pfadangaben. Den Inhalt dieser Datei kann man anschließend einfach über den Variablennamen ansprechen. Tippen Sie z. B.

print(beispiel)

ein, wird der Inhalt der eingelesenen Datei ausgegeben (Abbildung 11). Probieren Sie auch den Befehl view(beispiel) und statt der Ausgabe in der Console geht links oben ein Datenfenster auf, das die Daten sauber in Tabellenform

anzeigt.

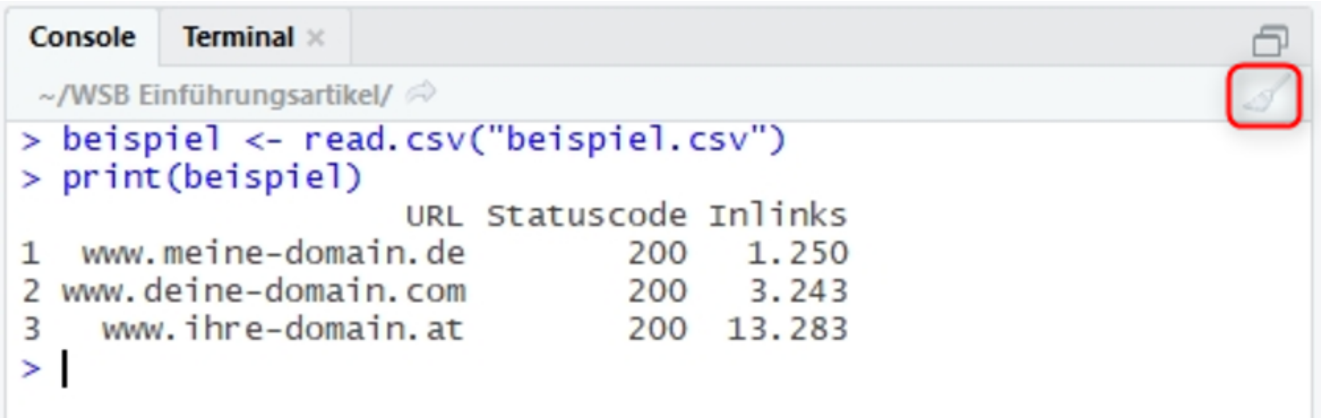


Abbildung 11: Dateien einlesen und anzeigen lassen
Geht es nicht nur um Datenanalysen, sondern werden Daten auch modifiziert oder neue Daten erzeugt, die man speichern möchte, geht auch das recht einfach mit dem Befehl write.table:

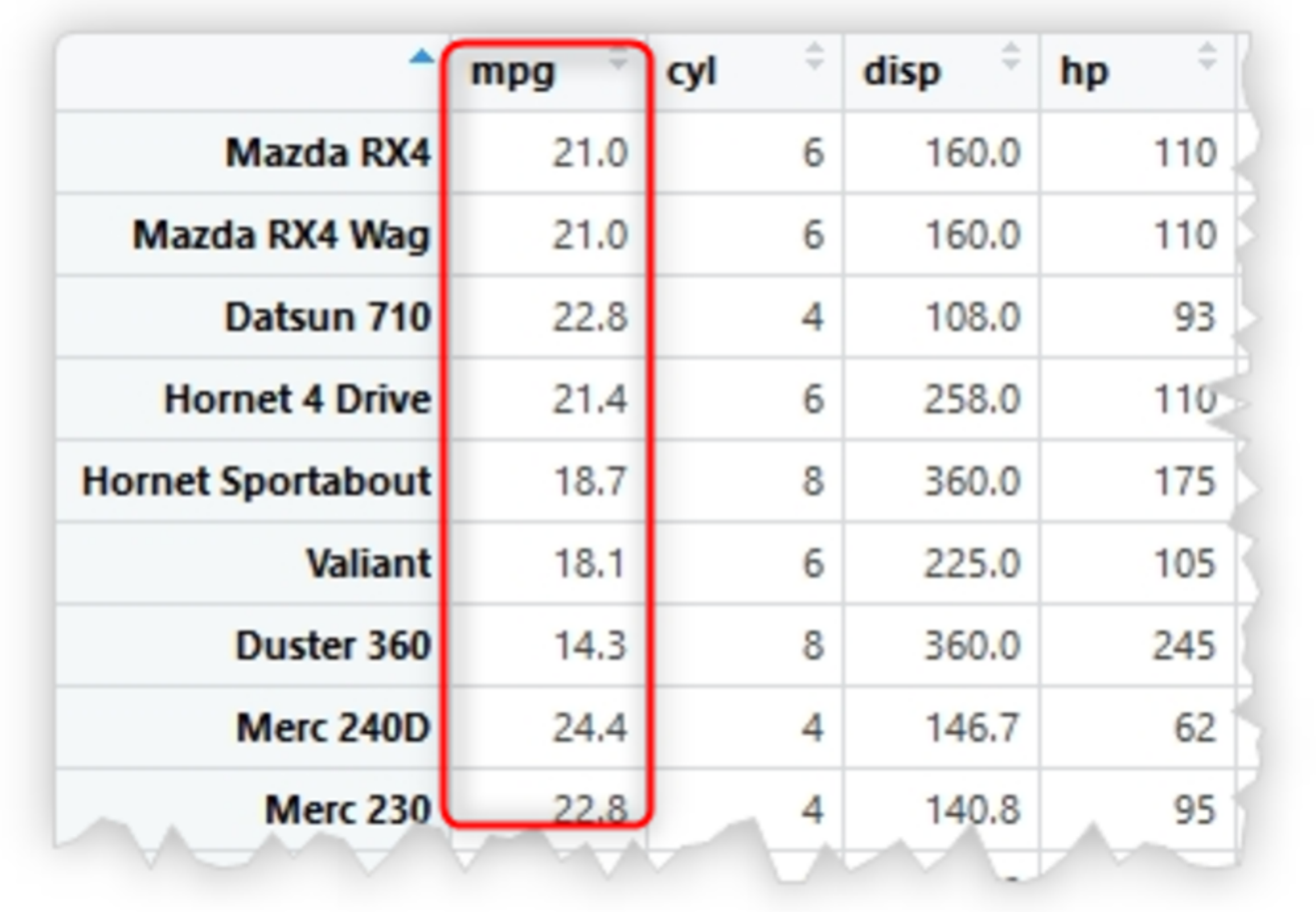
write.table(datensatz, „dateispeichername.csv“)

Möchten Sie z. B. die Daten des integrierten Beispiels „state.x77“ in eine CSV-Datei namens „test.csv“ in das aktive Arbeits-/Projektverzeichnis speichern, lautet der Befehl also:

write.table(state.x77, „test.csv“)

Abbildungen erzeugen

Natürlich lassen sich Grafiken in Excel recht einfach erzeugen. R hält jedoch deutlich mehr Möglichkeiten und nach Meinung vieler auch bessere Grafiken vor. Wer z. B. schon einmal versucht hat, aus einem Datensatz ein vernünftiges Histogramm zu erzeugen, wird sich über die fehlende Möglichkeit geärgert haben, die automatische Clusterung von Excel zu umgehen. Hat man stark abweichende Werte, sind Histogramme dort praktisch wegen der Spreizung nicht zu verwenden. In R stellt das Definieren von Clustern für die Histogrammsäulen kein Problem dar. Ebenso wenig wie die (befehlsorientierte) Formatierung einer Abbildung. Einmal definiert, lässt sie sich für Folgeabbildungen mit einer Zeile Code in immer demselben eigenen Design erzeugen.



	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	258.0	110
Hornet Sportabout	18.7	8	360.0	175
Valiant	18.1	6	225.0	105
Duster 360	14.3	8	360.0	245
Merc 240D	24.4	4	146.7	62
Merc 230	22.8	4	140.8	95

Abbildung 12: Anriss des mitgelieferten Datensatzes „mtcars“

Ein Histogramm erzeugt man in R mit dem einfachen Befehl „hist“ gefolgt von dem gewünschten Datensatznamen in Klammer dahinter. Eine der mitgelieferten Datenbeispiele ist die Datei mtcars. Mit dem nun schon bekannten Befehl

```
print(mtcars)
```

können Sie einen schnellen Blick darauf werfen. Möchte man nun den Benzinverbrauch aller dort gelisteten Modelle (erste Spalte, mpg, also miles per gallon) in einem Histogramm darstellen, benötigt man nur eine Befehlszeile:

```
hist(mtcars$mpg, col = „blue“)
```

Das Prinzip ist recht leicht ersichtlich. Aus den Daten „mtcars“ soll aus der Datenspalte „mpg“ ein Histogramm erzeugt werden (Befehl „hist()“), dessen Balken blau sind („blue“). Das Ergebnis zeigt Abbildung 13 auf der rechten Seite – und im Vergleich links das Histogramm, das Excel mit den gleichen

Daten erzeugt. R teilt die Datencluster statistisch vernünftiger auf und erstellt ein durchaus differenziertes Bild. Wenn Sie die Clusterbildung selbst beeinflussen möchten, weil Sie z. B. feinere Abstufungen brauchen oder einfach nur auch zeigen wollen, dass unter zehn Gallonen pro Meile eben kein Wert und damit keine Säule vorhanden ist, geht das über eine einfache Erweiterung des Befehls mit Angabe der gewünschten Intervalle „breaks“ (der Übersichtlichkeit halber wurde auf die Farbangabe hier verzichtet). Probieren Sie es einfach aus:

```
hist(mtcars$mpg, breaks = c(5,10,15,20,25,30,35))
```

Selbstverständlich könnte man die Intervalle auch mit einer eigenen Formel statt mit manuell eingegebenen Intervallen wie hier im Beispiel berechnen lassen.

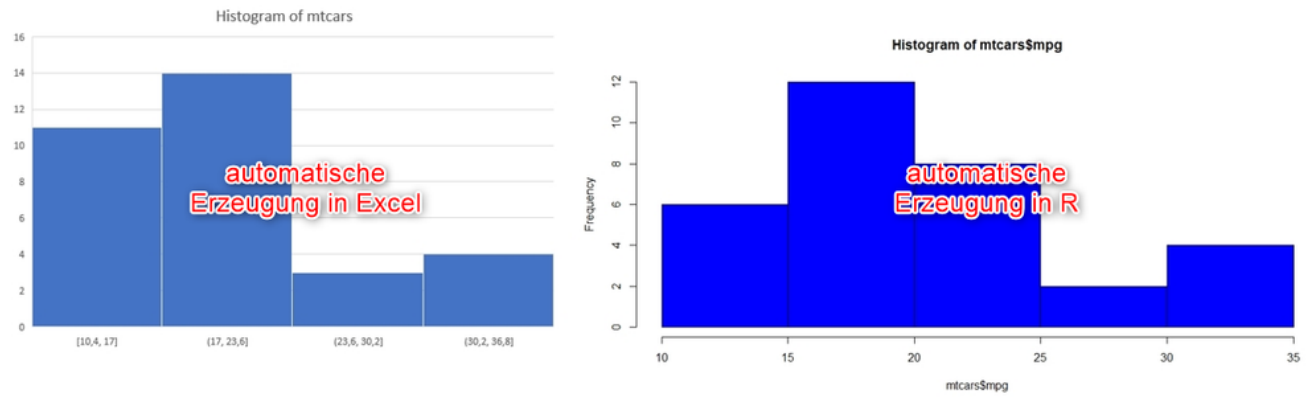


Abbildung 13: Vergleich eines unbearbeiteten Histogramms mit identischen Daten in Excel und R

Aber auch ein schneller optischer Blick auf strukturelle Eigenheiten von Datenreihen ist möglich. Dazu lässt man sich eine oder mehrere Datenreihen als Boxplot ausgeben. Zur Demonstration kann man ganz einfach erneut auf das bereits genutzte Datenset „mtcars“ zugreifen. Möchte man mehr über den Einfluss der Anzahl der Zylinder eines Motors auf den Verbrauch bzw. auf die Streuung der Daten wissen, setzt man beides einfach in Beziehung. Der Befehl dazu ist

```
boxplot(mpg ~ cyl, data=mtcars)
```

In Abbildung 12 ist ersichtlich, dass es zwei Spalten im

Datenset gibt, die mit „mpg“ (Miles per Gallon) und „cyl“ (Zylinder) überschrieben sind. Der Befehl „boxplot“ bezieht sich auf die angegeben Datenquelle, setzt die beiden Datenreihen dann ganz einfach in Beziehung und gibt Abbildung 14 aus.

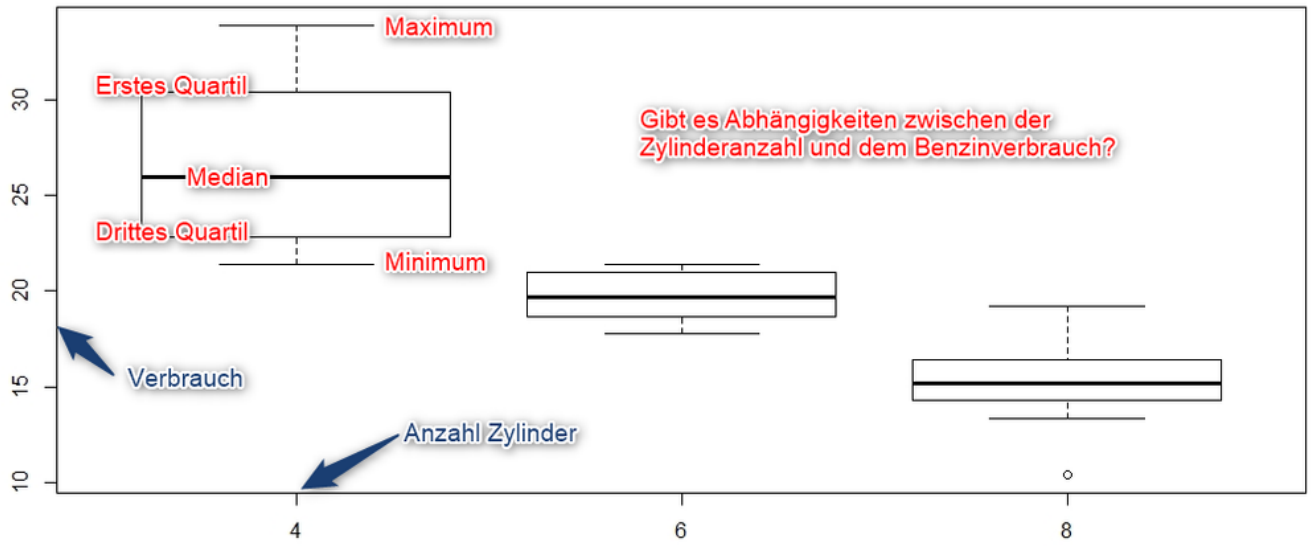


Abbildung 14: Statistische Kennwerte visualisieren

Dort erkennt man auf den ersten Blick, wie stark die Verbrauchsdaten des nach Zylinder geclusterten Verbrauchs streuen. Würde man statt mtcars auf einen eigenen eingelesenen Datensatz verweisen und zwei Spaltenüberschriften auswählen, müsste man den boxplot-Befehl einfach nur entsprechend anpassen und mehr nicht. Dazu wären in R nur zwei Befehlszeilen nötig – der zum Einlesen und der Befehl zur entsprechenden Ausgabe. Eine Sache von wenigen Sekunden.

Kleines Cheatsheet für statistische Funktionen

Durchschnitt/Mittelwert: mean()
Median (zentraler Wert): median()
Varianz: var()
Standardabweichung: sd()
Quantile: quantile()
Bandbreite von–bis: range()

> *„Vor der Verwendung von Daten ist es immer hilfreich, einen statistisch motivierten Blick auf die Struktur dieser Daten zu werfen.“*

Wer es eine Spur härter, aber keinesfalls komplexer haben möchte, kann auch problemlos mehrere Datenreihen miteinander korrelieren lassen bzw. optisch über Streudiagramme prüfen, ob es einzelne Abhängigkeiten bzw. Korrelationen gibt. Dies lässt sich gut bzw. schnell und einfach am Beispiel des Datensets „iris“ zeigen. Dort sind für drei Schwertlilienarten in Summe 150 Datensätze mit jeweils Länge und Breite der Kelchblätter (Sepal) und der Kronenblätter (Petal) hinterlegt. Mit

```
View(iris)
```

wird das Datenset angezeigt (Abbildung 15).

Geben Sie jetzt doch einfach den Befehl

```
plot(iris[-5])
```

ein. Sie erhalten zwölf Streudiagramme aller möglichen Kombinationen zwischen Kelchblättern und Kronenblättern nach Länge und Breite (Abbildung 16). In den beiden rot markierten Diagrammen lässt sich unmittelbar ein fast linearer Zusammenhang bezüglich Länge und Breite bei den Kelchblättern erkennen, während bei den Kronenblättern und zwischen Kronen- und Kelchblättern kein solcher offensichtlich ist.

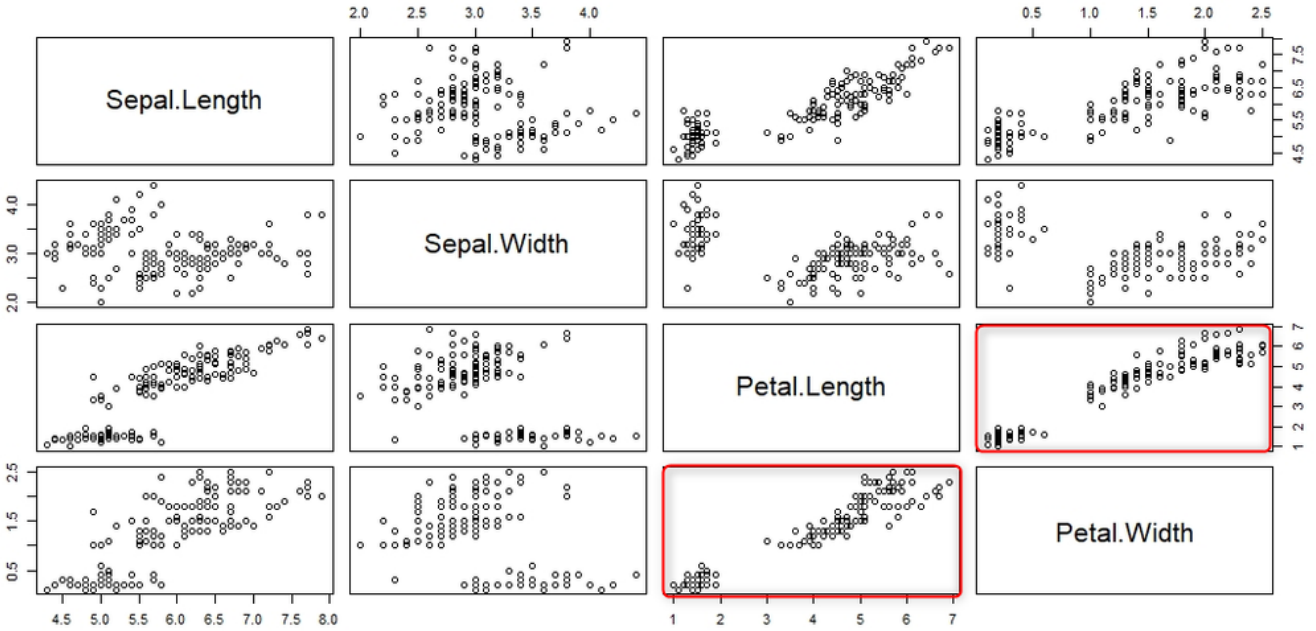


Abbildung 16: Korrelieren einzelne Daten miteinander?

Fazit

Tauschen Sie jetzt einfach geistig oder auch real in der Praxis die letzten Beispiele wie z. B. das florale Datenset mit echten Daten aus Ihrem Arbeitsumfeld aus – dann erkennen Sie zumindest in Ansätzen den Wert und die Transparenz, die R in wenigen Minuten bringen kann. Und das ganz ohne wirklich tiefergehende Programmierkenntnisse und nur mit dem Wissen über einige einzelne Befehlszeilen. Natürlich bleibt der Nutzen tatsächlich überschaubar, wenn Sie bei der Lektüre nur dieses Beitrags stehen bleiben. Aber stellen Sie sich doch mal vor, Sie steigen jetzt motiviert und in Erahnung der Kenntnis des Potenzials von R Stück für Stück tiefer ein. Sie haben noch nicht einmal ein halbes Prozent von dem ausprobiert, was R leisten kann, wenn Sie es richtig bespielen. Was nun am Ende mehr nützt – die Rationalisierungseffekte für einfacheres und schnelleres Arbeiten oder neue Erkenntnisse und bessere, wirklich datengestützte Entscheidungen –, bleibt Ihren Überlegungen und Ihrer Motivation überlassen.