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).

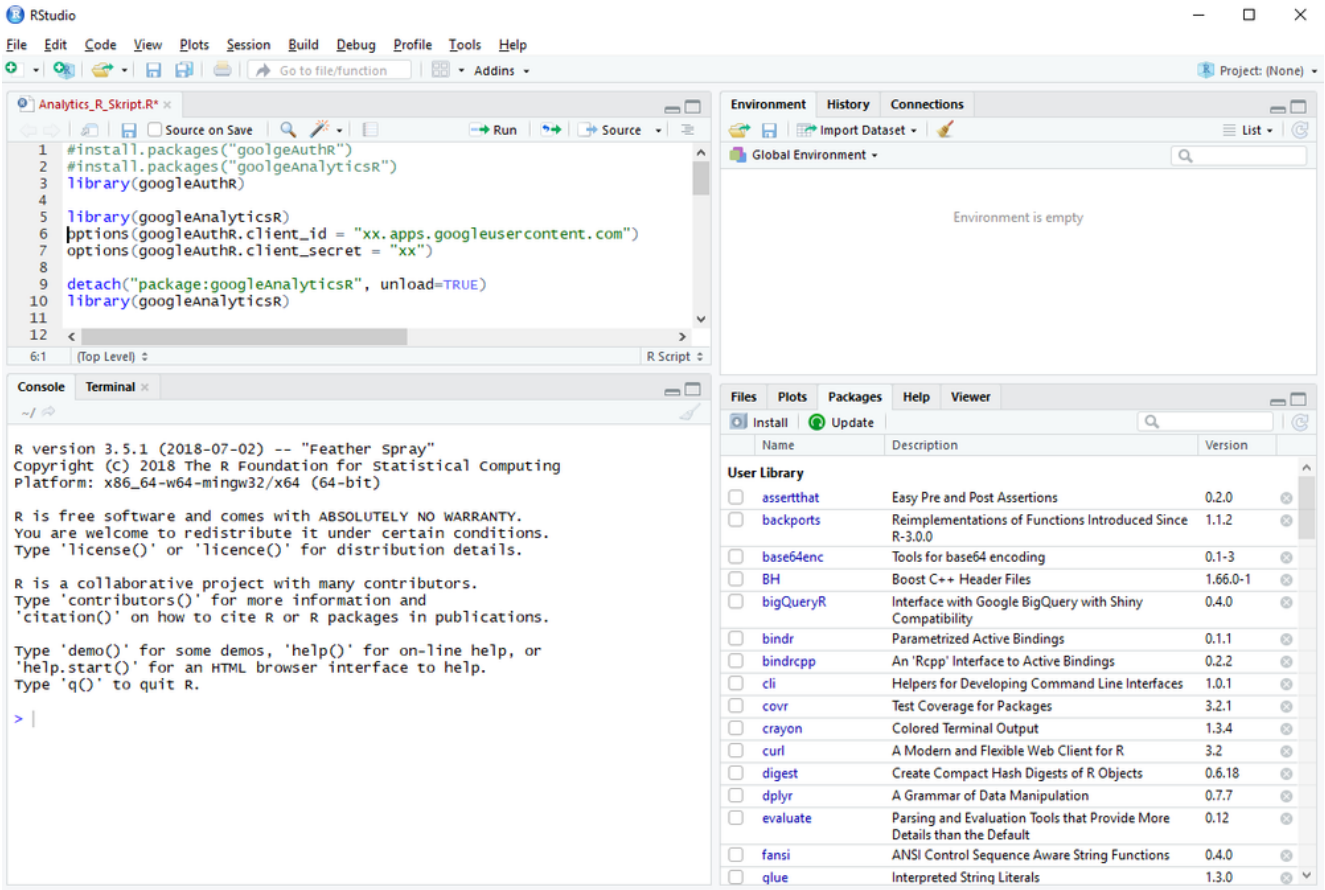


Abb. 1: RStudio „bettet“ die Software R benutzerfreundlich ein und macht die Bedienung komfortabler

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).

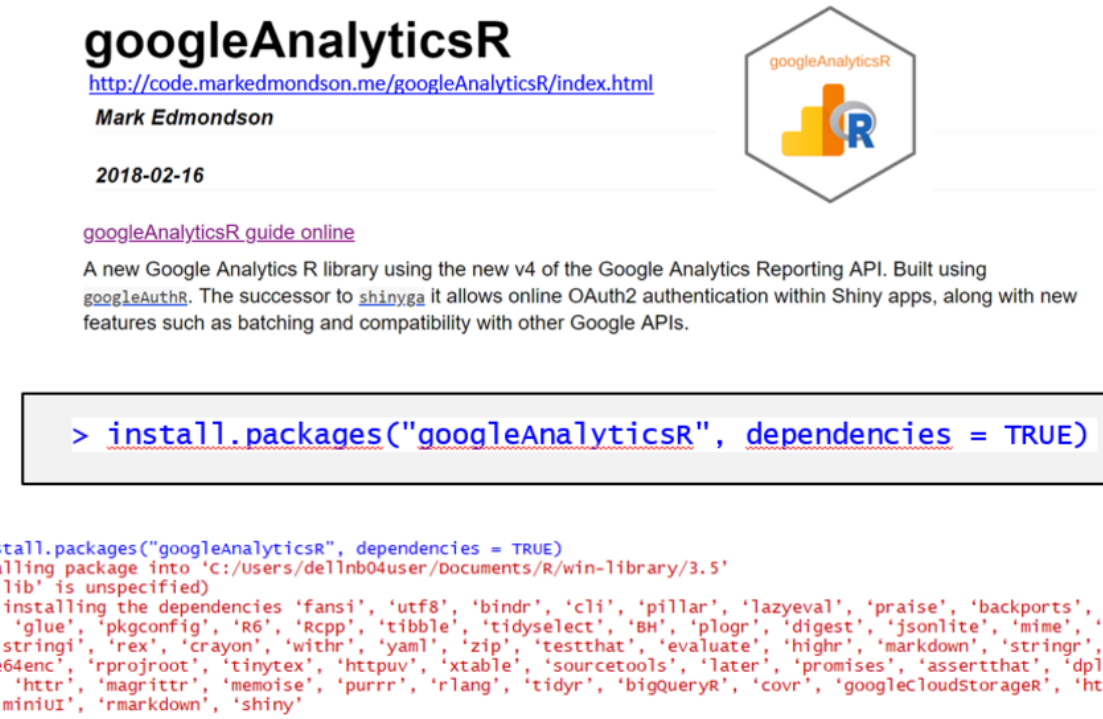


Abb. 2: Einfache Installation von googleAnalyticsR nebst weiteren Paketen direkt in R

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).

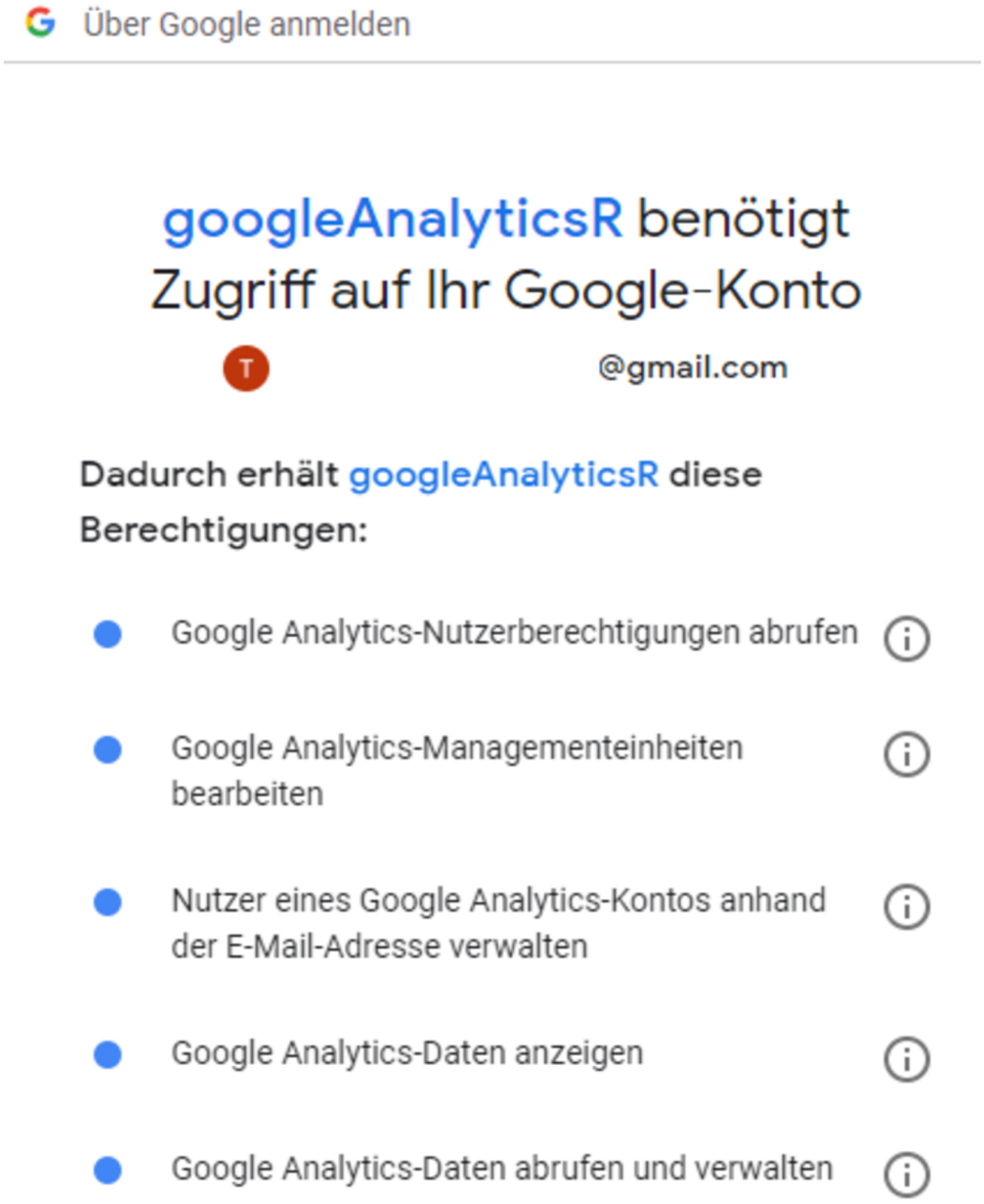


Abb. 3: Zugriff des R-Pakets auf Google-Konto

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.

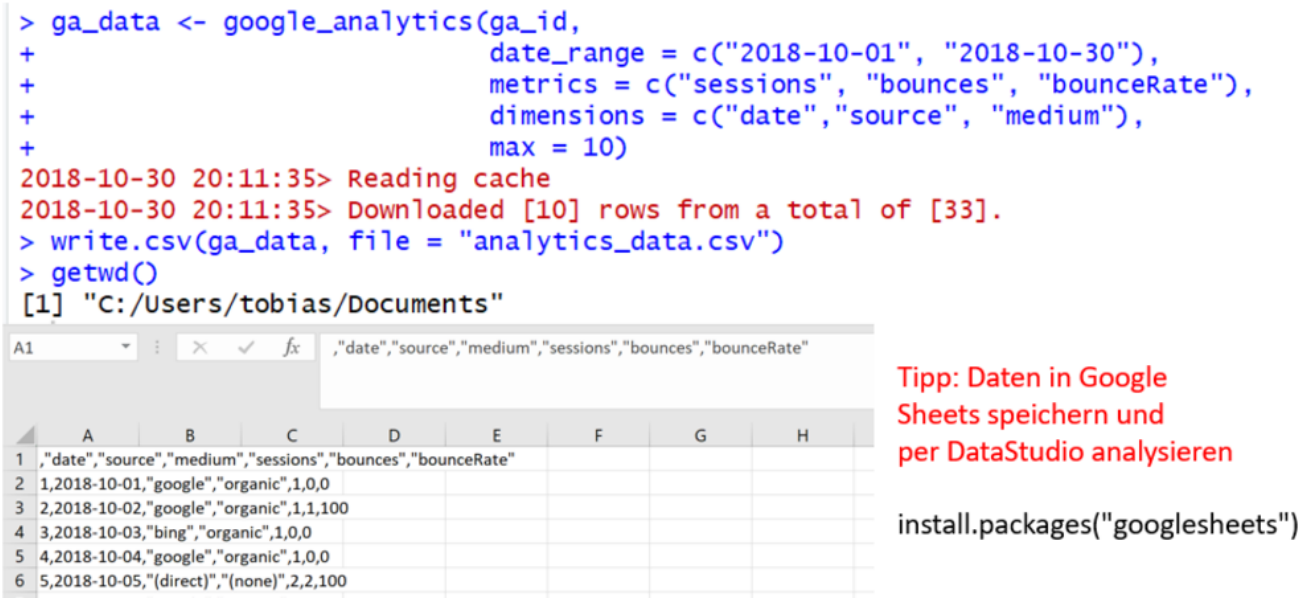


Abb. 5: Daten aus der API in R analysieren und als .csv speichern

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).

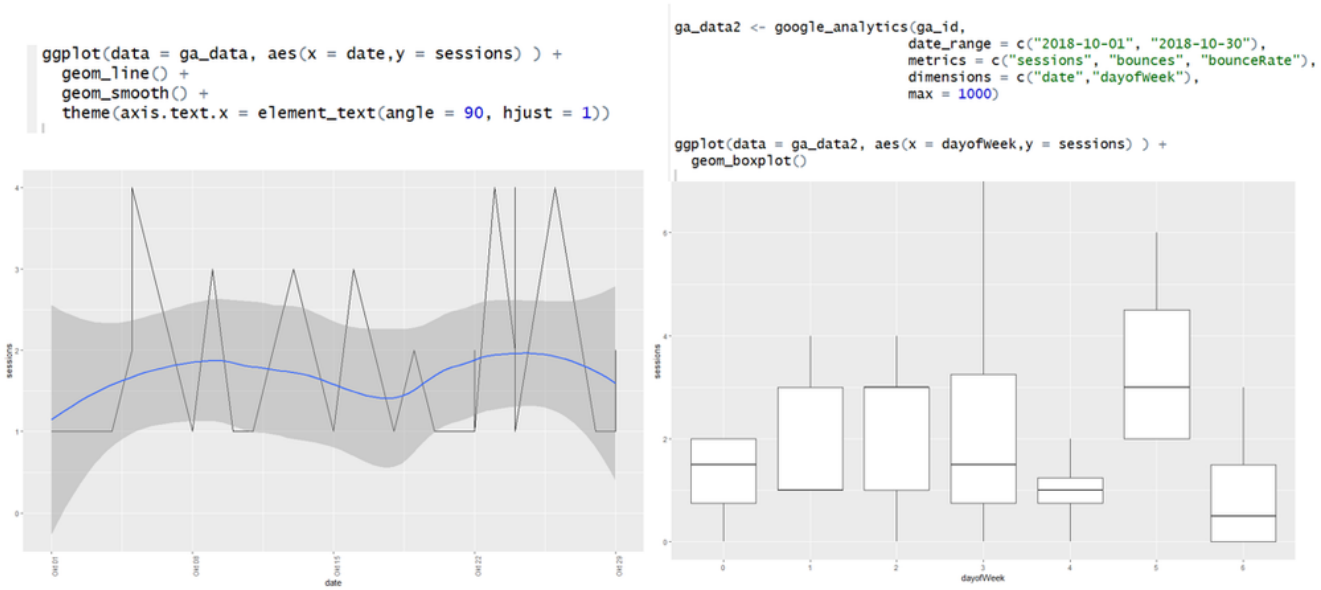


Abb. 6: Datenvisualisierungen mit dem Paket ggplot2

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.

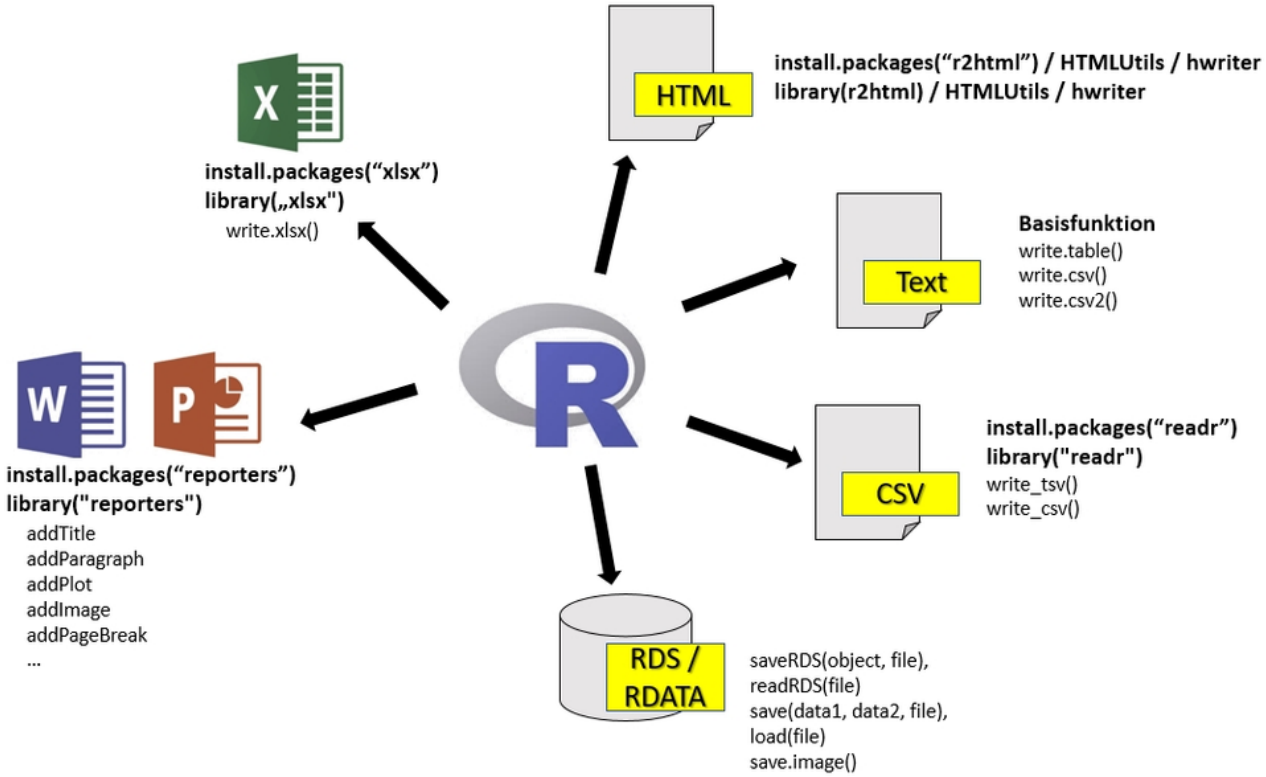


Abbildung 1: Auszug für gängige Ausgabe- bzw. Exportformate in R

„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.

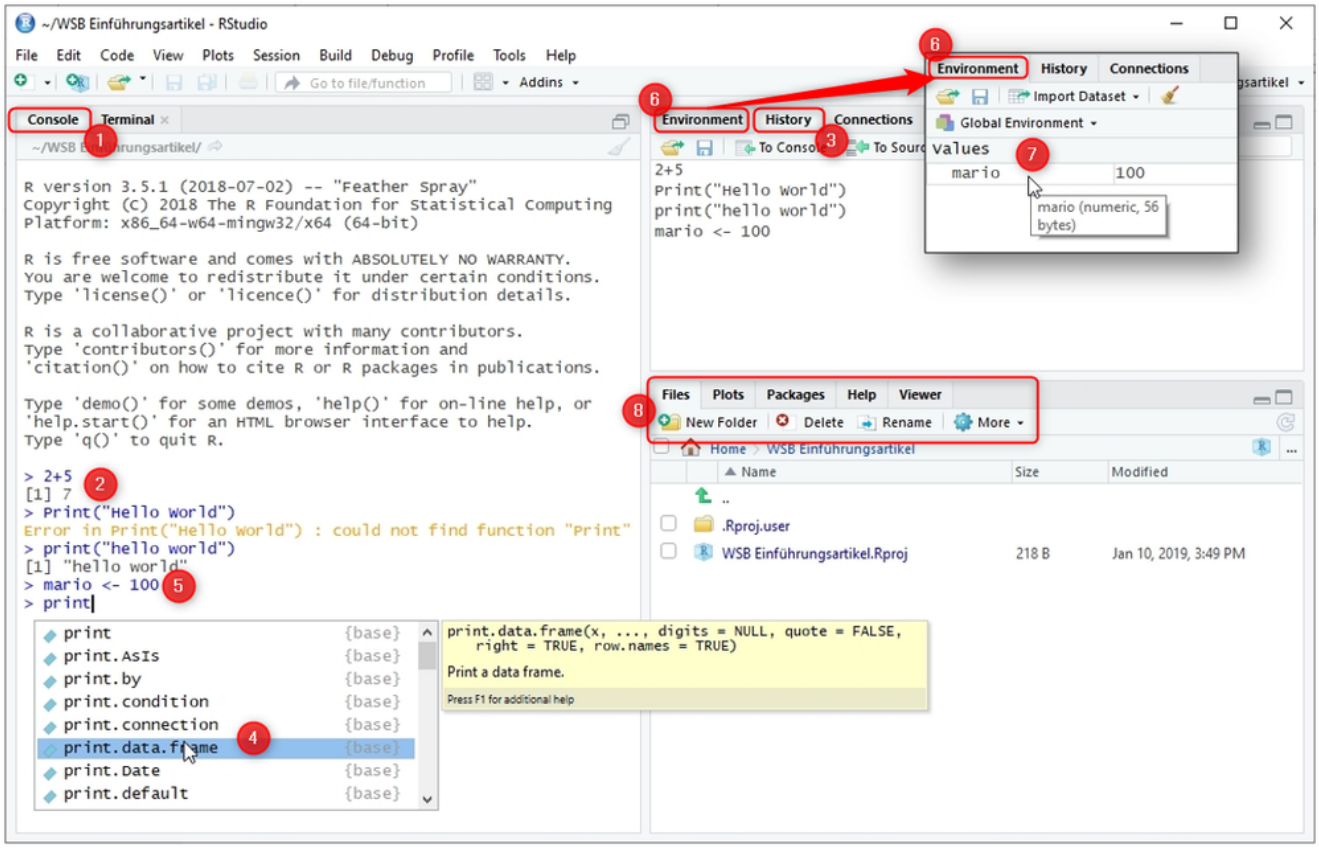


Abbildung 2: Der Startbildschirm von RStudio

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.

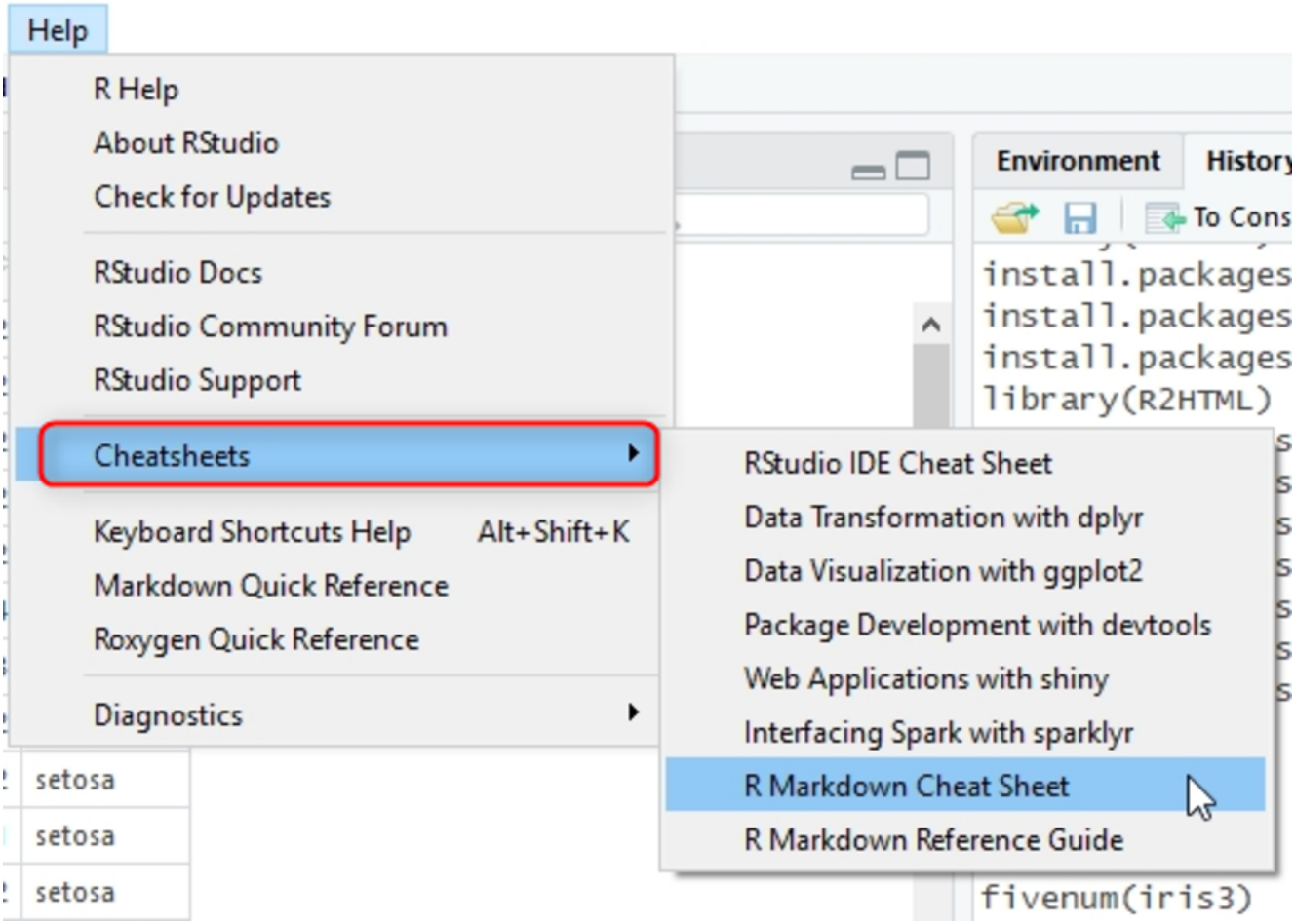


Abbildung 3: In RStudio findet man unter dem Menüpunkt „Help“ nützliche Cheatsheets

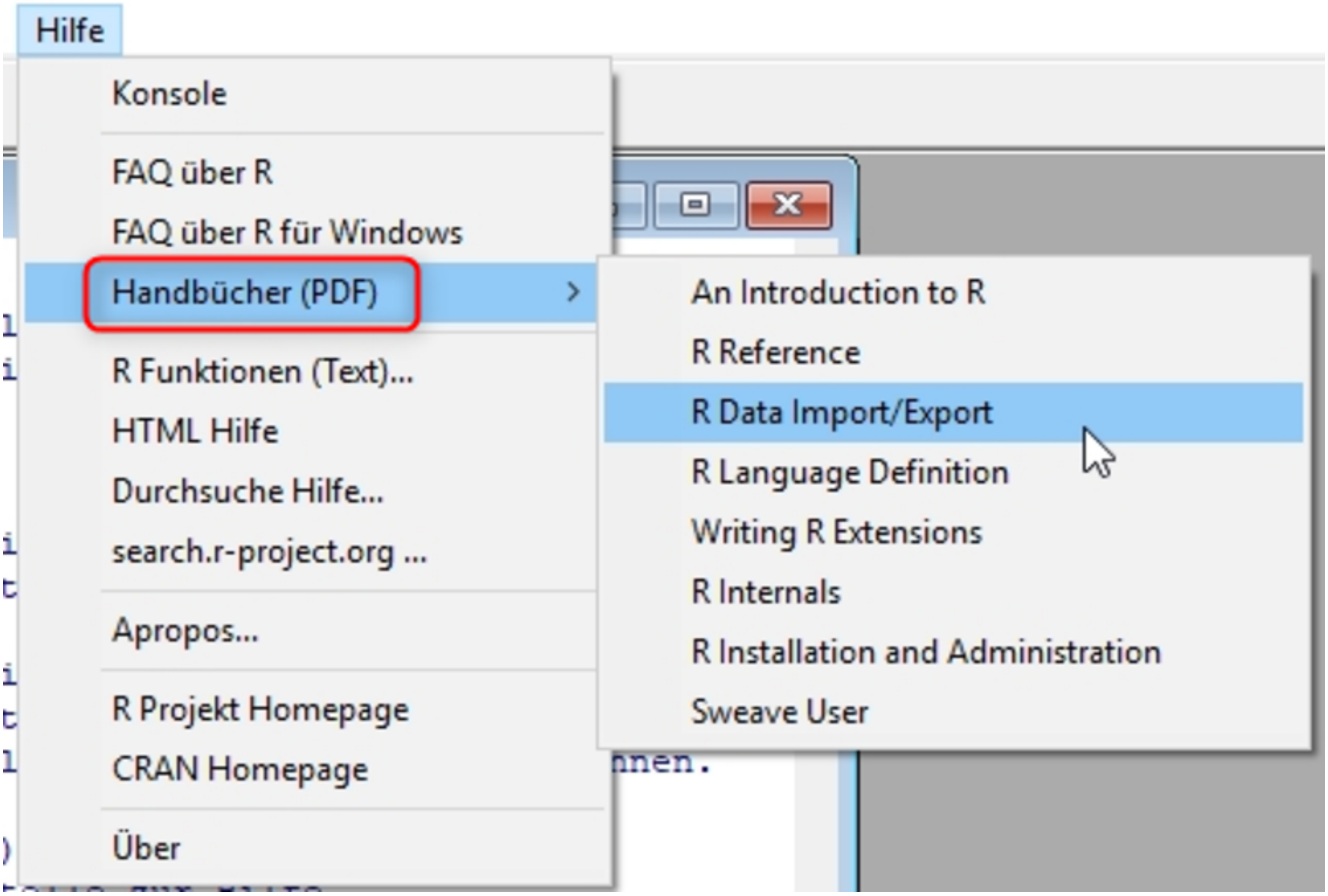


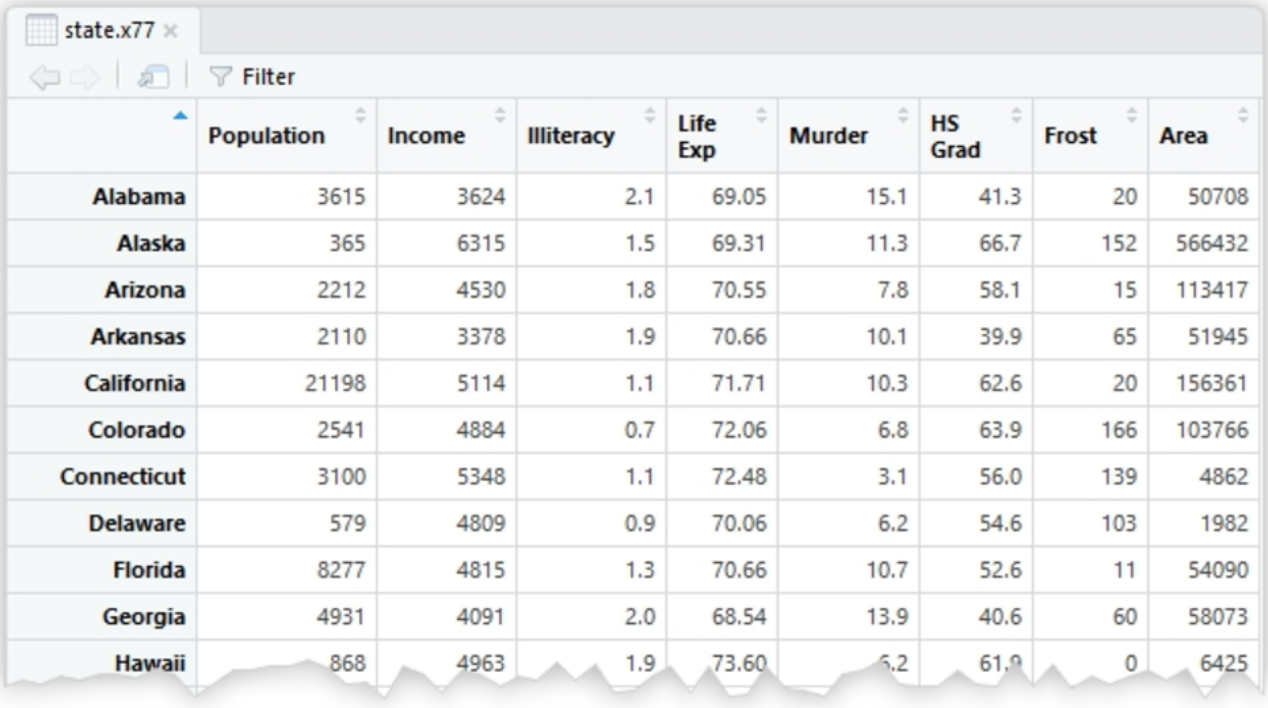
Abbildung 4: In R selbst (nicht in RStudio) sind umfangreiche PDF-Handbücher hinterlegt (englisch)

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.

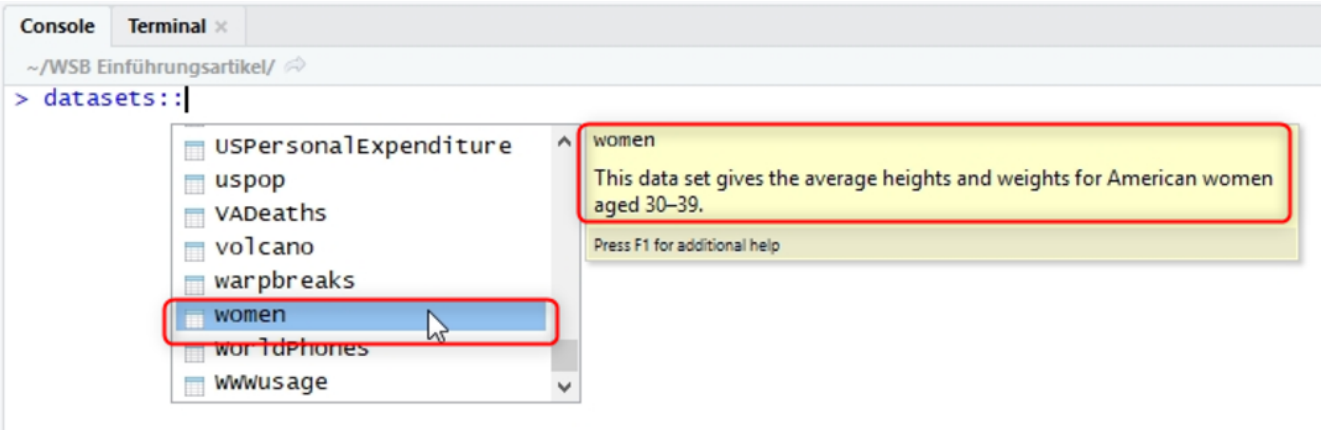


Abbildung 7: Automatisch mitgelieferte Datensätze zum Ausprobieren mit Inhaltsbeschreibung

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.

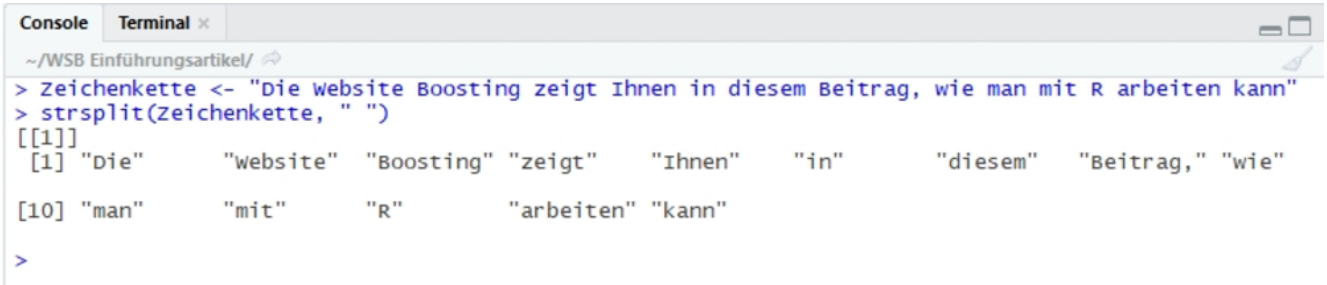


Abbildung 8: Mit einem einzigen Befehl Texte in Wörter zerlegen

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)

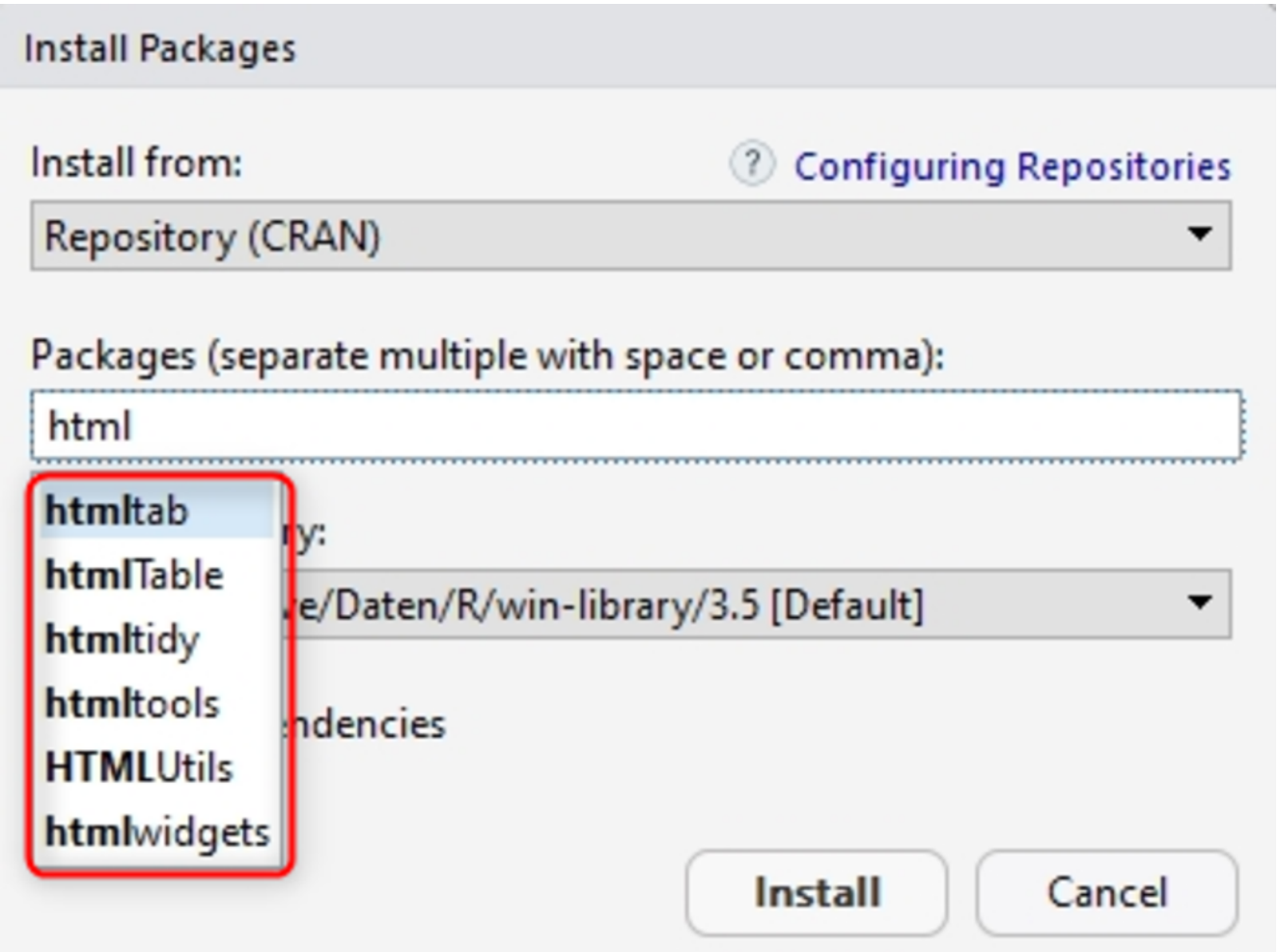


Abbildung 9: Erweiterungen lassen sich sehr einfach installieren

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.

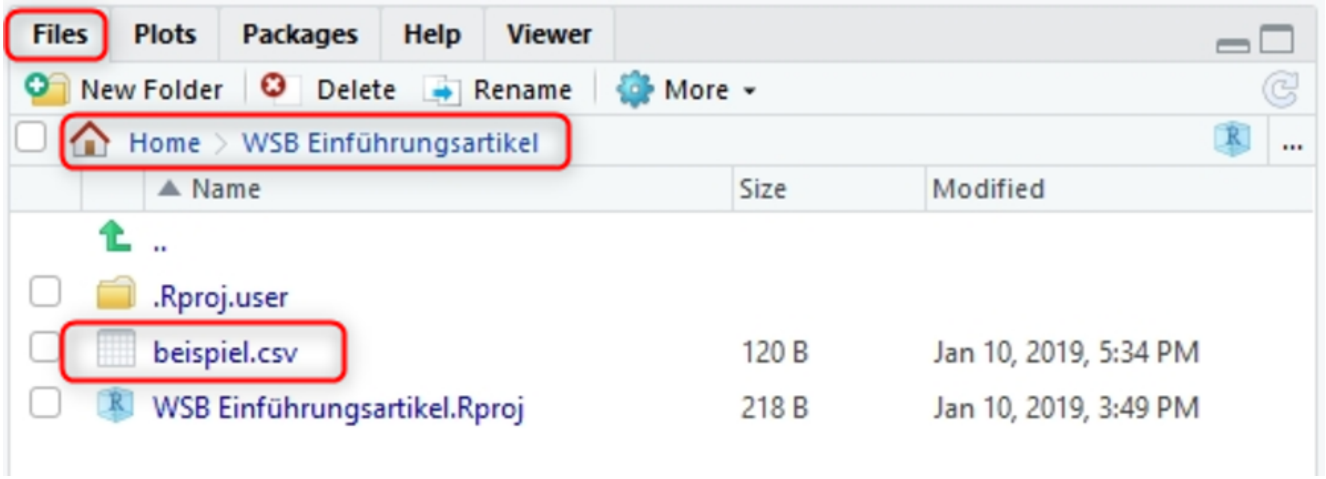


Abbildung 10: Legen Sie über ein neues Projekt einen Speicherort zum Testen an
Über den Befehl

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.

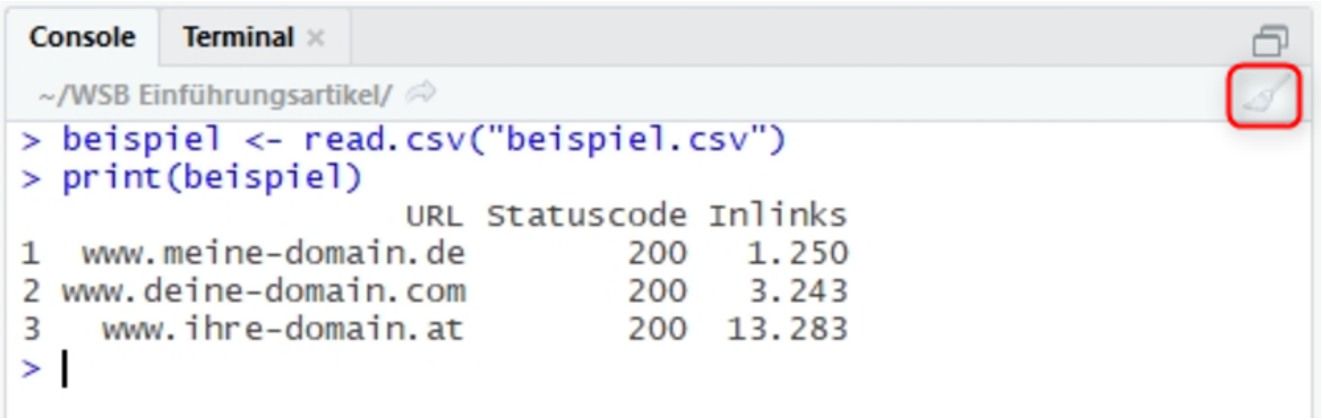


Abbildung 11: Dateien einlesen und anzeigen lassen
Geht es nicht nur um Datenanalysen, sondern werden Daten auch modifiziert oder neue Daten erzeugt, die man speichern möchte, geht auch das recht einfach mit dem Befehl write.table:

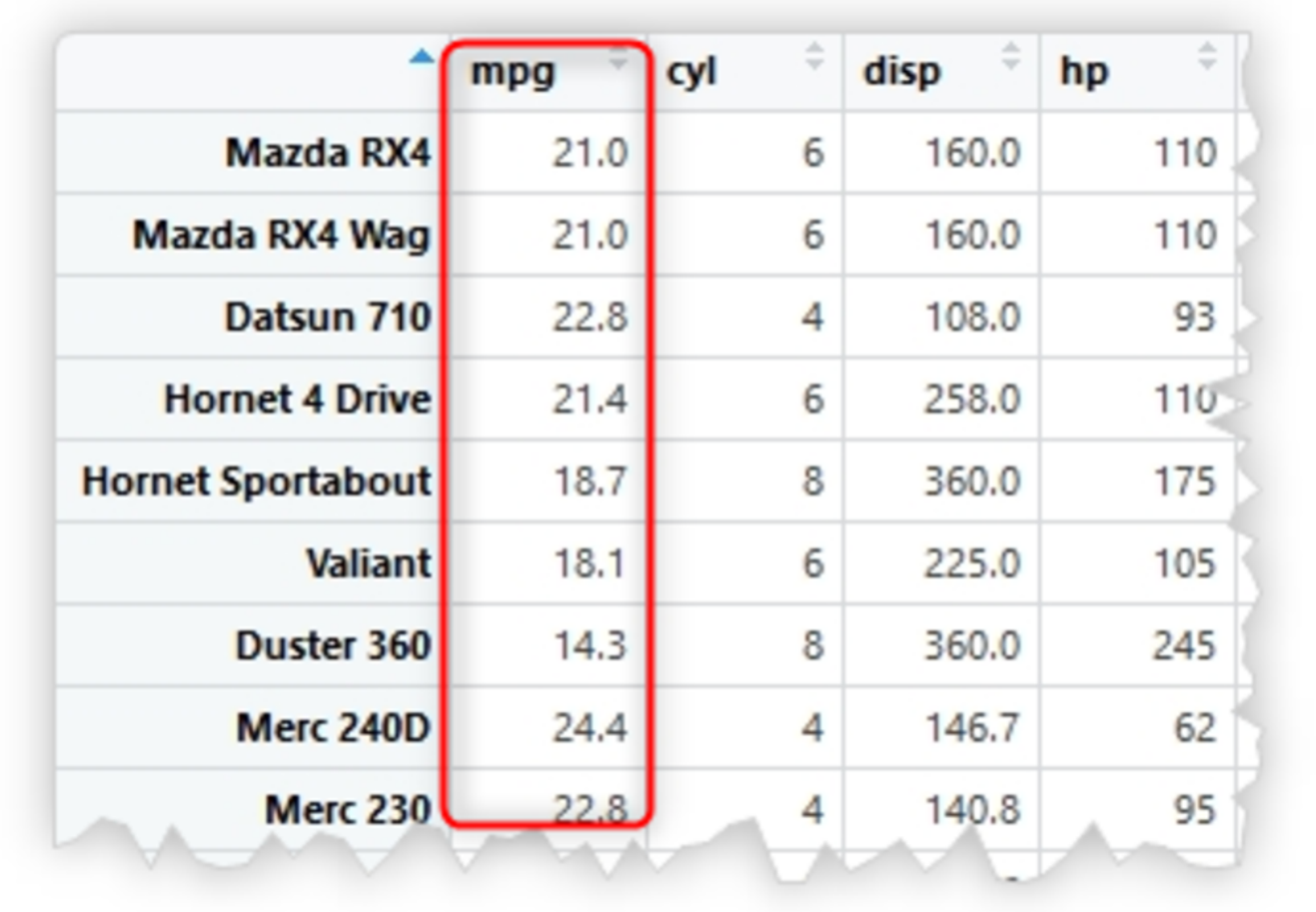
write.table(datensatz, „dateispeichername.csv“)

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.

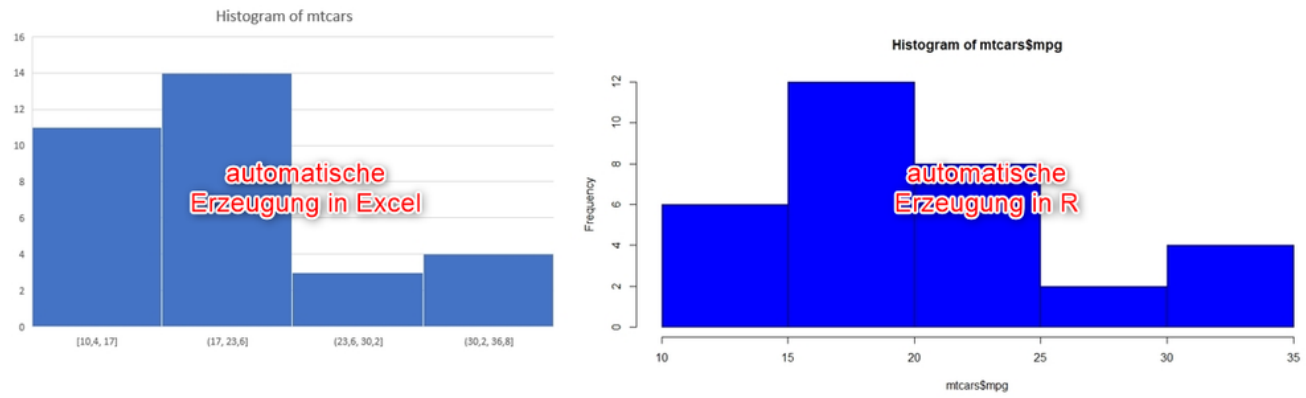


Abbildung 13: Vergleich eines unbearbeiteten Histogramms mit identischen Daten in Excel und R

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.

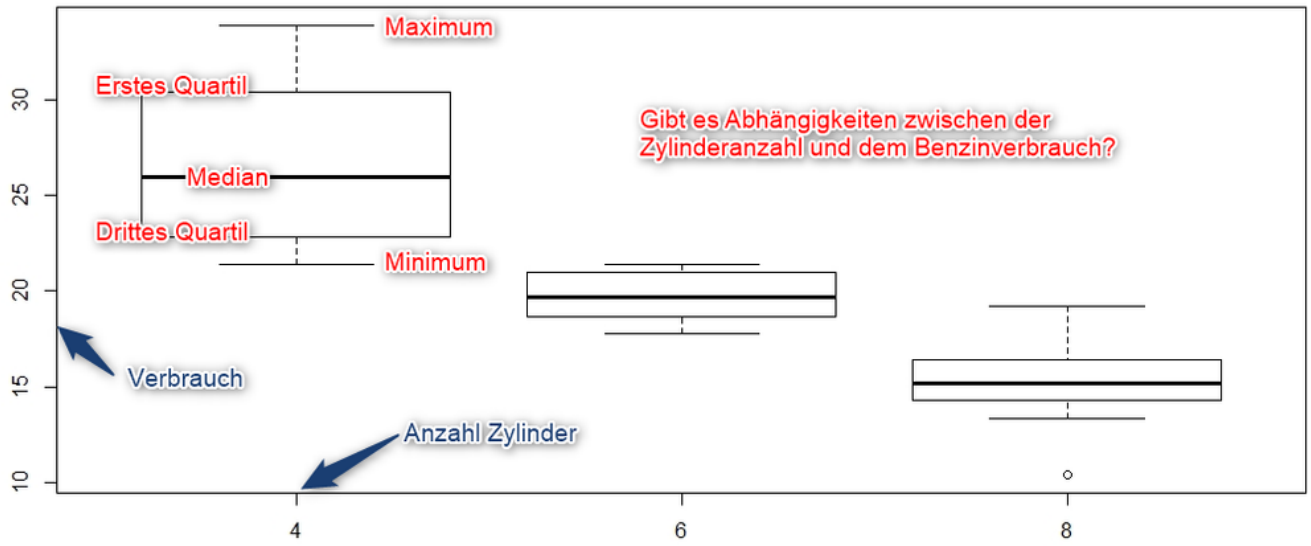


Abbildung 14: Statistische Kennwerte visualisieren

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.

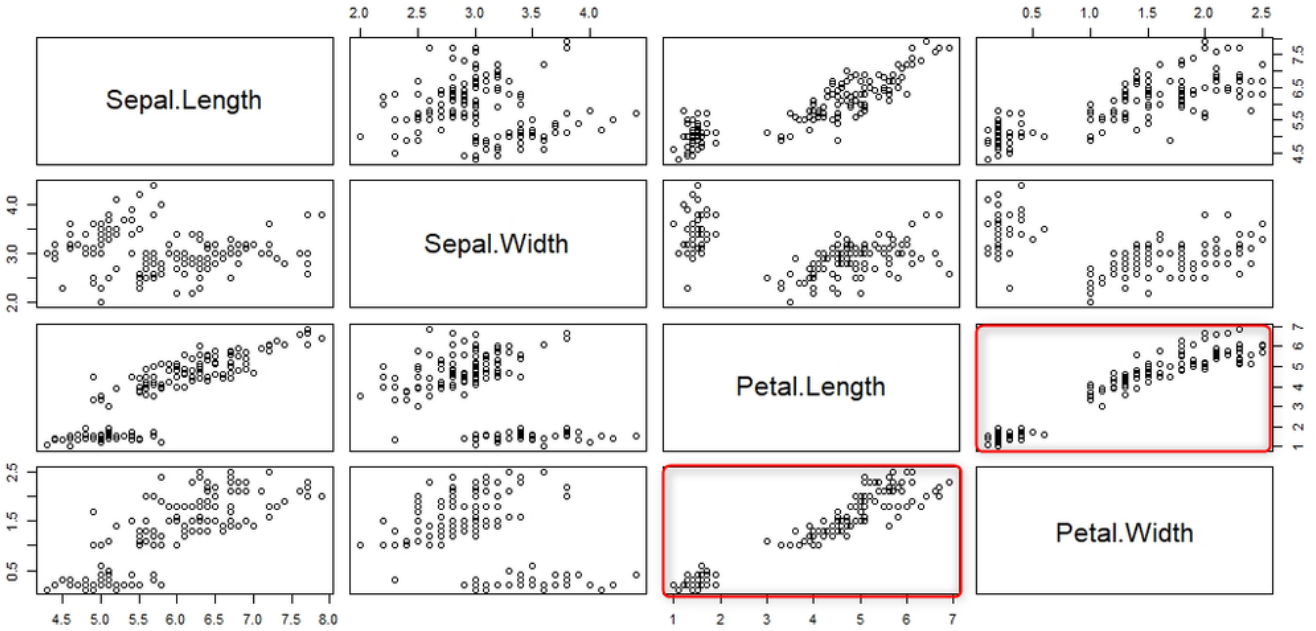


Abbildung 16: Korrelieren einzelne Daten miteinander?

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.

Vitamin D: Datenanbindung über R an Google Schnittstellen

Vitamin D:Datenanbindung über R an Google

Schnittstellen – websiteboosting.com

R ist nicht nur Open Source (damit kostenlos) und anpassungsfähig durch vielfältige Pakete, sondern auch offen für andere (technische) Verbindungen. Diese Verbindungen, vulgo API, ermöglichen, flexibel Daten an der Ursprungsquelle abzufragen, entsprechend weiterzuverarbeiten und ggf. modifiziert…
R ist nicht nur Open Source (damit kostenlos) und anpassungsfähig durch vielfältige Pakete, sondern auch offen für andere (technische) Verbindungen. Diese Verbindungen, vulgo API, ermöglichen, flexibel Daten an der Ursprungsquelle abzufragen, entsprechend weiterzuverarbeiten und ggf. modifiziert wieder zurückzuliefern. Google bietet eine Vielzahl von API, die in R eingebunden eine enorme Individualisierung und Automatisierung von analytischen Prozessen und Visualisierungen ermöglichen. Die Excel-Alternative Google Sheets in Kombination mit Google Data Studio ist bereits eine unkomplizierte Lösung für vielfältige Visualisierungen. Gepaart mit der statistischen Leistung von R eröffnet sich durch die weitere Kombination mit API nochmals ein deutlicher Zugewinn an Flexibilität! Tobias Aubele zeigt, wie Sie diese Schnittstellen nutzen können.

R wird kontinuierlich von einer großen Community weltweit weiterentwickelt, Gleiches gilt für die Dienste von Google wie Maps & Co. Unglaubliche Mengen an Daten liegen in diversen Datensilos und „sehnen“ sich danach, verarbeitet bzw. interpretiert zu werden und einen Nutzen zu stiften. Durch die Vielzahl an Silos und steigenden Datenmengen bedarf es einer effektiven und effizienten Verarbeitung, damit letztlich genügend Zeit für die Interpretation der Daten und das Ableiten von Maßnahmen übrig bleibt. Obwohl ggf. sehr zeitaufwendig, wird in Fachbereichen gerne Microsoft Excel eingesetzt, da dort Daten mittels Formeln und Funktion aus diversen Quellen flexibel zusammengeführt und verarbeitet werden können. Leider sind Daten teilweise veraltet, ehe sie letztlich über manuelle Zusammenfassungen und Verdichtungen

für den eigentlichen Nutzungszweck zur Verfügung stehen. Sofern direkt API (Programmierschnittstellen) genutzt werden, können aktuelle Daten unmittelbar am Ursprung abgeholt und zurückgeliefert werden und bieten langfristig ggf. ein hohes Potenzial für automatisierte Abläufe.

Programmcode online verfügbar!

Damit Sie Codezeilen nicht einzeln abtippen müssen, sind zwei der Skripte aus diesem Beitrag direkt online für Sie zum Download hinterlegt:
Das Skript zum Knowlede Graph unter www.websiteboosting.com/code/r-knowledge
Das Skript für Google Sheets unter www.websiteboosting.com/code/r-sheets

Die API – aktuelle Daten direkt vom Ursprung

Über die Google API können diverse Google-Dienste direkt über R angesprochen werden. Nach einer Registrierung unter console.cloud.google.com kann die gewünschten API, wie bspw. die des Google Knowledge Graph, einzeln aktiviert und unmittelbar getestet bzw. genutzt werden (siehe Abb. 1).

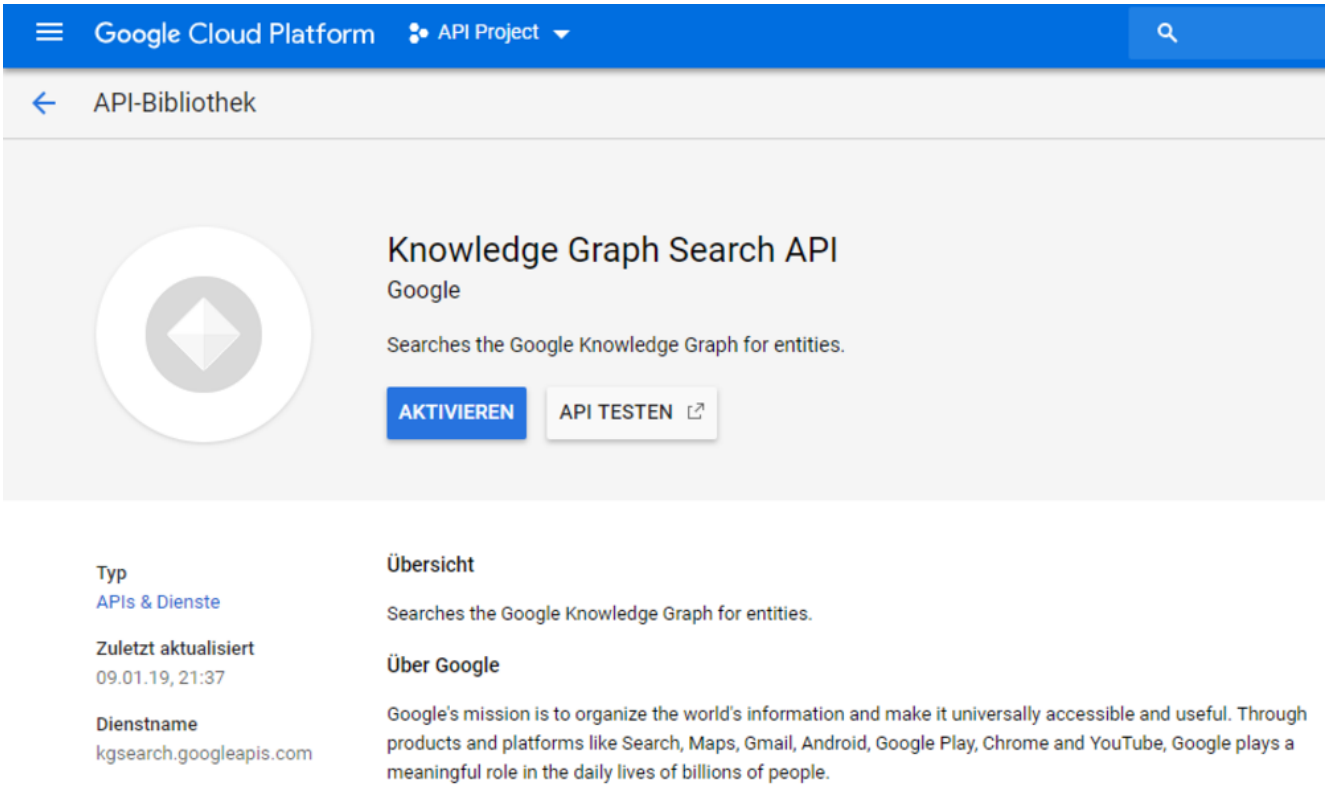


Abb. 1: Aktivierung einer Google API

Weiterhin können anschließend die notwendigen Anmeldedaten wie ein API-Schlüssel oder eine OAuth-Client-ID erstellt werden (siehe Abb. 2). Damit wird sichergestellt, dass der Datenzugriff nur von autorisierten Nutzern ermöglicht wird.

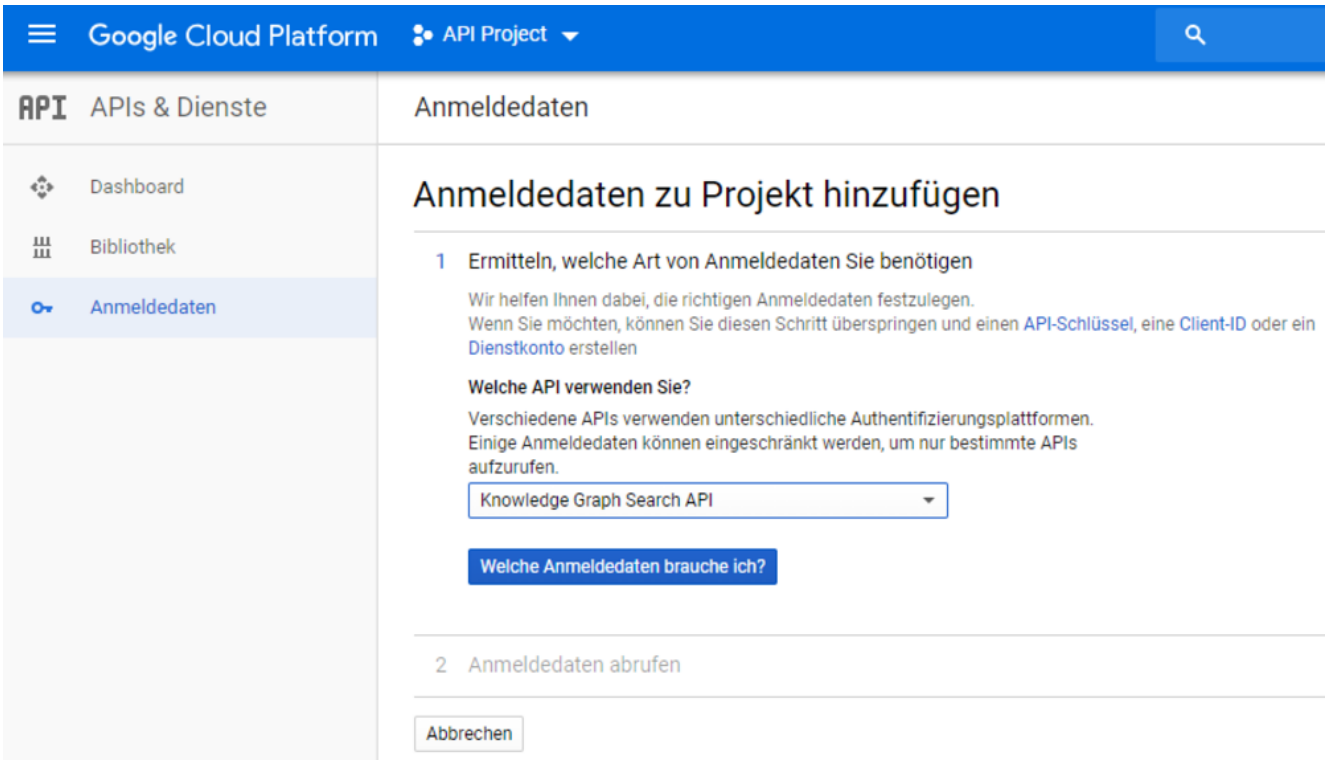


Abb. 2: Erstellung von gesicherten Anmeldemechanismen

Mittels der API werden die Daten direkt aus der Datenbank von

Google abgerufen. Es stehen damit nicht nur aktuelle, sondern auch ggf. erweiterte Daten zur Verfügung. Das heißt, Informationen über Albert Einstein, welche im Graphen am rechten Bildschirmrand angezeigt werden (siehe Abb. 3), stehen direkt zur weiteren Verarbeitung zur Verfügung. Die API ermöglicht die Extraktion weiterer Daten, sollten mit dem Suchbegriff mehrere Entitäten (Personen, Objekte) in Verbindung gebracht werden wie bspw. die Person Hans Albert Einstein.

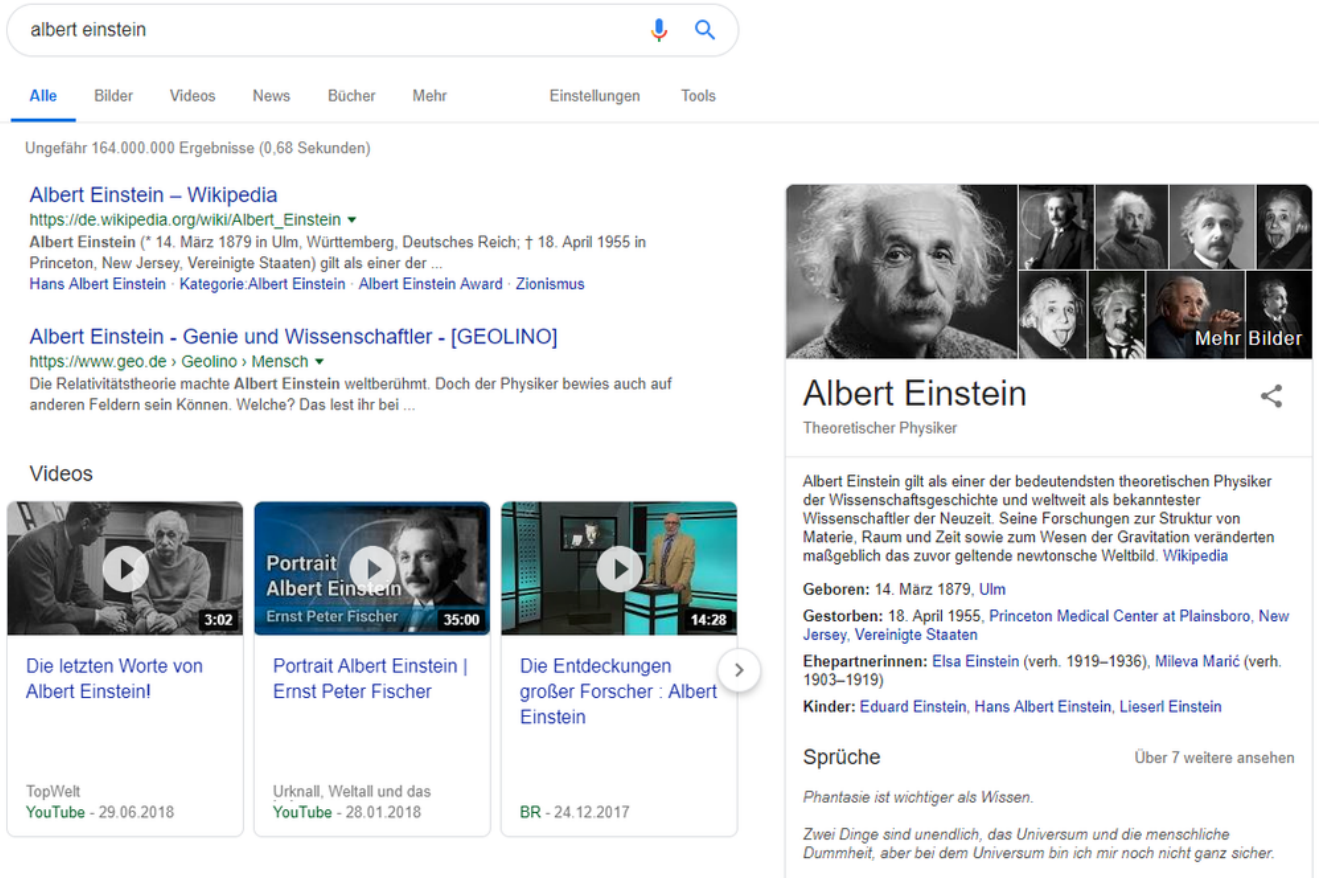


Abb. 3: Google Knowledge Graph von Albert Einstein

Das zur Datenextraktion notwendige R-Paket lautet *GoogleKnowledgeGraphR*, entwickelt von Daniel Schmeh (zur Dokumentation siehe das PDF unter: [einfach.st/r4kgraph)](einfach.st/r4kgraph). Mittels des individuellen API-Keys (Generierung siehe Abb. 2) werden mit dem Befehl

```
gkg_raw
```

die gewünschten Informationen in diversen Sprachen abgerufen. Das heißt, es stehen neben der Beschreibung unmittelbar die

URL des Bildes, die URL des Eintrages sowie Informationen zu den Nutzungsrechten (dennoch prüfen!) zur Verfügung (siehe Abb. 4). Ohne die jeweilige Suchabfrage bei Google starten zu müssen, können die Ergebnisse direkt in R abgerufen und weiterverarbeitet werden. Dies kann zu einer weiteren Vereinfachung und damit zur Zeitersparnis führen.

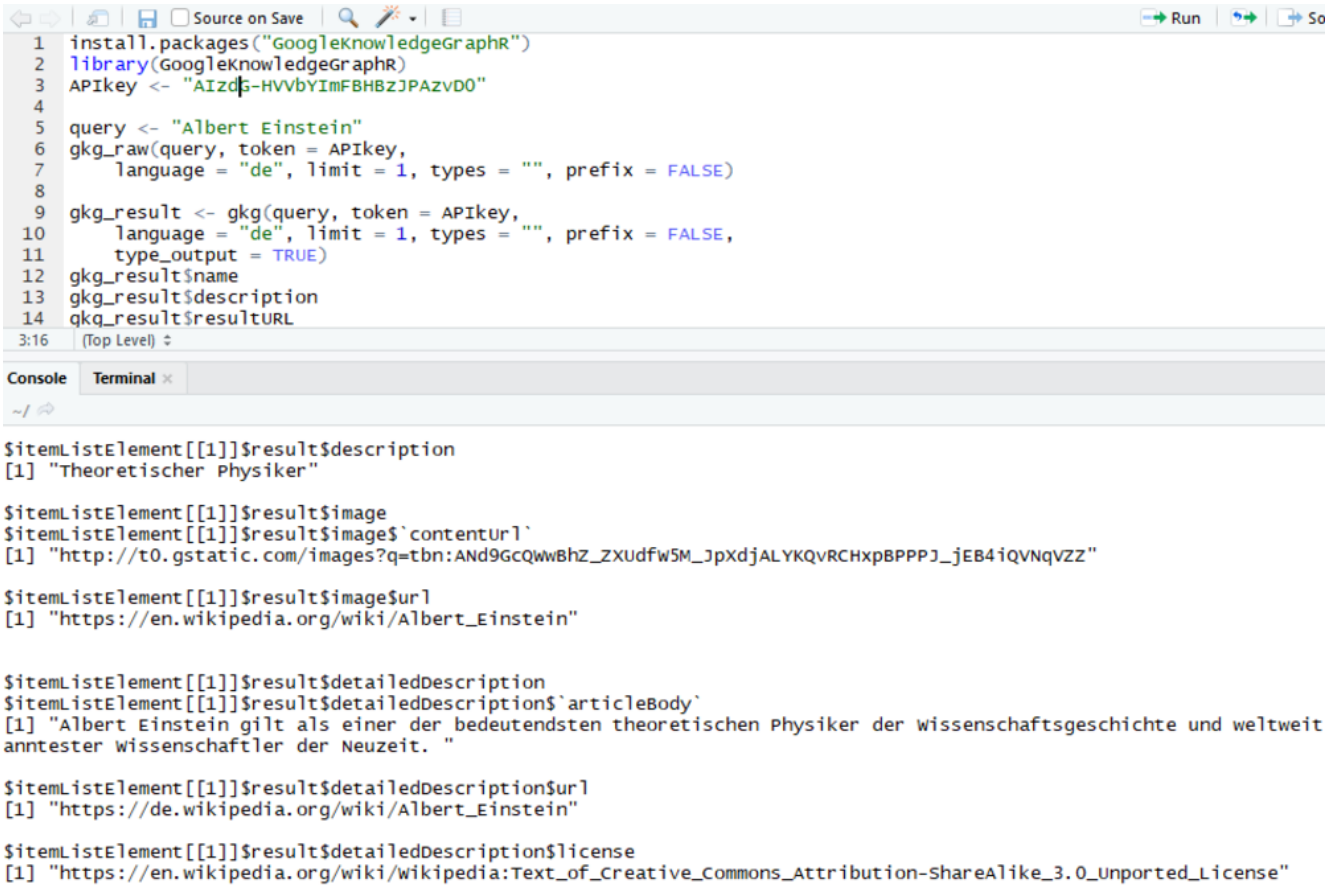


Abb. 4: Aufruf der Knowledge Graph API über R

Ähnlich diesem Prozess erfolgt die generelle Nutzung von Google API (mit R). Es bedarf eines API-Schlüssels bzw. einer Autorisierung sowie eines korrespondierenden Programms (hier R-Paket). In der Praxis können nahezu alle Google API, wie bspw. die Google Translator API oder die Google Maps/Places API, abgefragt werden. Vereinzelt sind die Services kostenpflichtig mit einem flexiblen Abrechnungsmodell wie bspw. auf Basis der Anzahl an API-Abfragen. Damit könnten Textübersetzungen bzw. Abfragen zu Distanzen, Öffnungszeiten, Adressen etc. von Restaurants und anderen Google-Places-Einträgen vollzogen werden. Die entsprechenden R Packages heißen *googleway* und *googleLanguageR*.

Kollaboration mit Google Sheets

Das Besondere an Google Sheets ist der weltweite Zugriff, die große Rechenleistung sowie die gleichzeitige Zusammenarbeit mit mehreren Personen. Weiterhin können Daten aus diversen Quellen zusammengeführt (vergleichbare Excel-Funktionen SVERWEIS bzw. INDEX/VERGLEICH) und letztlich ein Bericht sehr flexibel gestaltet, individualisiert und geteilt werden. Daher erscheint es sinnvoll, die Möglichkeiten von Google Sheets sowie ggf. der nachgelagerten Visualisierung mit Google Data Studio zu nutzen.

Daten in R können an Google Sheets bzw. von Google Sheets übermittelt werden. Hierzu sind die Pakete *googlesheets* (PDF zur Dokumentation siehe: einfach.st/r4sheets) bzw. *googleAuthR* notwendig bzw. hilfreich. Nach der Installation des Paketes über den Befehl

install.packages(“googlesheets“)(siehe Abb. 6)

erfolgt die Freigabe bzw. Authentifizierung des R-Pakets mit Google Drive über

gs_auth() (siehe Abb. 5).

tidyverse-googlesheets benötigt Zugriff auf Ihr Google-Konto

T [redacted]@gmail.com

Dadurch erhält tidyverse-googlesheets diese Berechtigungen:

Google Drive-Dateien aufrufen, bearbeiten, erstellen und löschen

Tabellen in Google Drive aufrufen, bearbeiten, erstellen und löschen

Überlegen Sie sich gut, ob Sie tidyverse-googlesheets vertrauen

Eventuell teilen Sie vertrauliche Informationen mit dieser Website oder App.In den Nutzungsbedingungen und der Datenschutzerklärung von tidyverse-googlesheets erfahren Sie alles zum Umgang mit Ihren Daten. In Ihrem Google-Konto können Sie die Zugriffsberechtigungen jederzeit einsehen oder entfernen.

Weitere Informationen zu den Risiken

Abbrechen

Zulassen

Abb. 5. Autorisierung des R-Pakets
Nach erfolgreicher Authentifizierung kann mit dem Befehl

gd_user()

der aktuelle Nutzer sowie der Berechtigungsstatus abgefragt werden. Der Befehl

gs_new

ermöglicht, ein neues Sheet anzulegen sowie den Namen bzw. Arbeitsblattnamen zu vergeben. Ab diesem Zeitpunkt können Daten eingegeben bzw. das Sheet genutzt werden. Der Befehl

gs_delete

löscht das Sheet entsprechend. Das jeweilige Sheet muss vor einer Aktion immer exakt adressiert werden, wozu diverse Möglichkeiten zur Verfügung stehen (siehe Dokumentation). Die exakte Art ist über die URL via

gs_url (siehe Abb. 6),

sofern Namen einzigartig sind, auch hilfsweise über den Titel (Befehl: *gs_title*). Soll das Sheet im Browser aufgerufen werden, muss der Befehl

gs_browse

mit dem Namen des Sheets (im Beispiel in der Variable gs gespeichert) und dem Blattnamen (relative Nummer bzw. absoluter Name) ausgeführt werden.

```
#install.packages("googlesheets") #Einmalig installieren
#install.packages("goolgeAuthR")  #Einmalig installieren
library(googlesheets)
library(googleAuthR)
gs_auth()
#gd_user()                        #Abfrage aktueller Nutzer

gs_new(title = "Analytics", ws_title = "Daten", row_extent = NULL, col_extent = NULL)

gs <- gs_title("Analytics")
gs <- gs_url("https://docs.google.com/spreadsheets/d/15cOt1RsXUxmB9RyrqXWb9yPiFoKxQk5ubA8YbUiKhAE")
gs_browse(gs, ws = 1)            #Google Sheets im Browser öffnen

gs_delete(gs, verbose = TRUE)
```

Abb. 6: Erstellung eines Google Sheets innerhalb von R

Google Sheets mit Daten füllen

Prinzipiell können auch Daten aus Google Analytics in R abgerufen und analysiert werden (siehe den ersten Teil dieses Beitrags in der letzten Ausgabe #53). Dem Google Sheet „Analytics“ kann über den Befehl

gs_ws_new

ein neues Arbeitsblatt mit dem Namen Oktober hinzugefügt werden (siehe Abb. 7) und anschließend mit den Google-Analytics-Daten befüllt werden (Abruf der Daten über den Befehl *google_analytics* aus dem R-Paket googleAnalyticsR). Die neuen Informationen sollen ab der Position (anchor) A1 zeilenweise (byrow = FALSE) eingefügt werden. Sollten dann im späteren Verlauf weitere Zeilen hinzugefügt werden (und dabei die alten Daten nicht überschrieben werden), kann ein Arbeitsblatt ausgelesen (Befehl *gs_read*), die aktuelle Anzahl an benutzten Zeilen festgestellt (nrow) und anschließend über die R-Funktion *paste0* ein neuer Start ermittelt (im Beispiel Zelle A38) und an die Variable gs_*position* übermittelt werden. Ein weiterer Aufruf der Funktion *gs_edit_cells* fügt die Daten demnach nicht wieder an der Position A1 ein, sondern nutzt als initiale Startposition den Inhalt der Variable gs_position.

```
18  library(googleAnalyticsR)
19  ga_auth()
20  ga_id <- "xxx"
21  ga_data <- google_analytics(ga_id,
22                              date_range = c("2018-10-01", "2018-10-31"),
23                              metrics = c("sessions", "bounces"),
24                              dimensions = c("date","landingpagepath"))
25
26  gs_ws_new(gs, ws_title = "Oktober", row_extent = 1000, col_extent = 26, verbose = TRUE)
27  gs <- gs_title("Analytics")
28  gs_edit_cells(gs, ws = "Oktober", input = ga_data, anchor = "A1", byrow = FALSE,
29                col_names = NULL, trim = FALSE, verbose = TRUE)
30
31  gs_lesen <- gs_read(gs, ws = 2, range = NULL, verbose = TRUE)
32  gs_position <- paste0("A",nrow(gs_lesen)+2)
33  gs_position
34
35  gs_edit_cells(gs, ws = 2, input = ga_data, anchor = gs_position, byrow = FALSE,
36                col_names = NULL, trim = FALSE, verbose = TRUE)
```

Abb. 7: Daten an Google Sheets aus R übertragen

Highlight: spezifische Google-Sheets-Funktionen

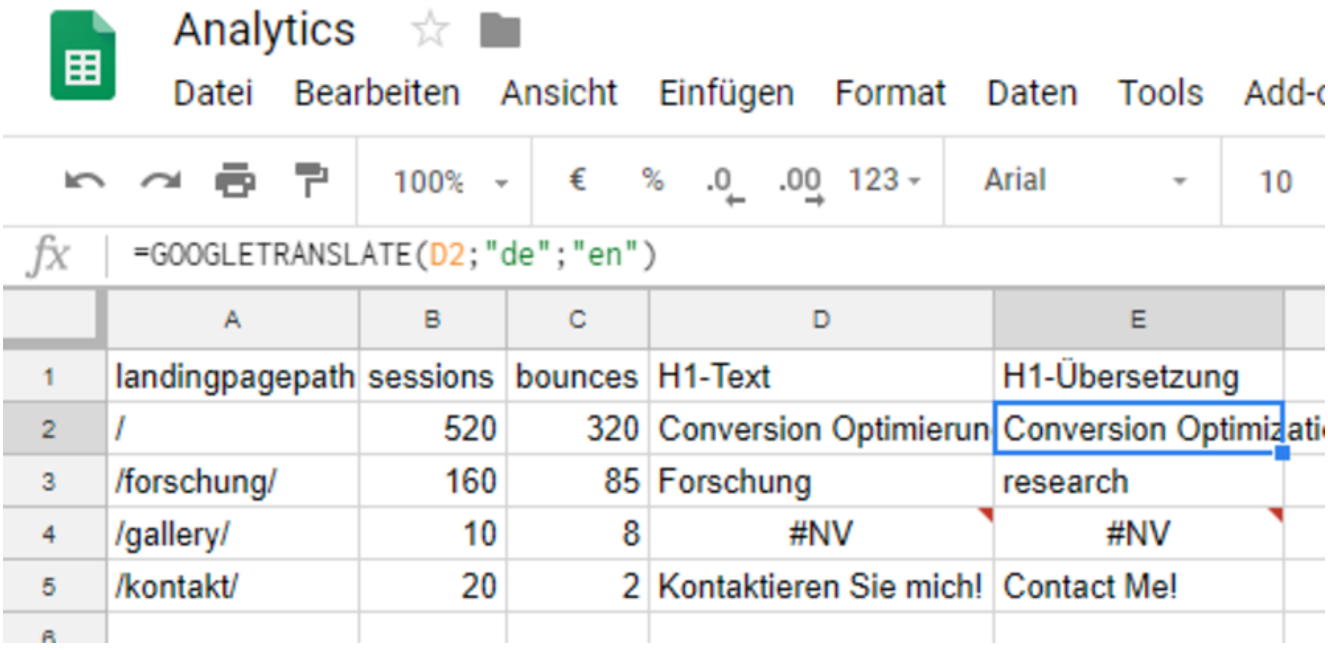
Mittels Austausches von Information an Google Sheets (lesen und schreiben) kann ein flexibler Datenaustausch mit R erfolgen. Ein weiterer Vorteil von Sheets ist die Mannigfaltigkeit spezieller Funktionen. Über die Funktion

```
=IMPORTXML(VERKETTEN("http://www.meineSeite.de";A2);"//h1")
```

kann eine XPath-Abfrage generiert werden, welche Elemente aus beliebigen Webseiten direkt ausliest (hier der H1-Text; siehe Abb. 8). Mittels einer weiteren Funktion

```
=GOOGLETRANSLATE(D2;"de";"en")
```

wird der Inhalt einer Zelle in die englische Sprache übersetzt. Die Funktion Verketten ermöglicht in diesem Zusammenhang, URL-Fragmente aus Google Analytics zu einer kompletten URL zusammenzuführen. Die Daten in Google Sheets können im weiteren Verlauf (manuell) beliebig verändert und anschließend wieder in R eingelesen/ weiterverarbeitet oder mit einer Visualisierung in Google Data Studio verbunden werden.



	A	B	C	D	E
1	landingpagepath	sessions	bounces	H1-Text	H1-Übersetzung
2	/	520	320	Conversion Optimierun	Conversion Optimizati
3	/forschung/	160	85	Forschung	research
4	/gallery/	10	8	#NV	#NV
5	/kontakt/	20	2	Kontaktieren Sie mich!	Contact Me!

Abb. 8: Nutzung von Google-Sheets-spezifischen Funktionen

Google Data Studio hat diverse Verbindungen zu Datenquellen, was für die Analyse und Aufbereitung eine hohe Flexibilität bietet. Weiterhin können beliebige Google-Tabellen als Datenquelle initiiert und anschließend in die Berichte integriert werden – bspw. jeweils als eigene Seiten. Die Anbindung geschieht über zwei Schritte. Erstens muss über den Google Tabellen-Connector das jeweilige Tabellenblatt gewählt und somit als Datenquelle definiert werden (siehe Abb. 9). Zweitens müssten anschließend die damit geschaffenen Datenverbindungen mit einem spezifischen Bericht verbunden und die jeweiligen Datenfelder ausgewählt werden. Zur Visualisierung stehen diverse Möglichkeiten zur Verfügung (siehe Abb. 10).

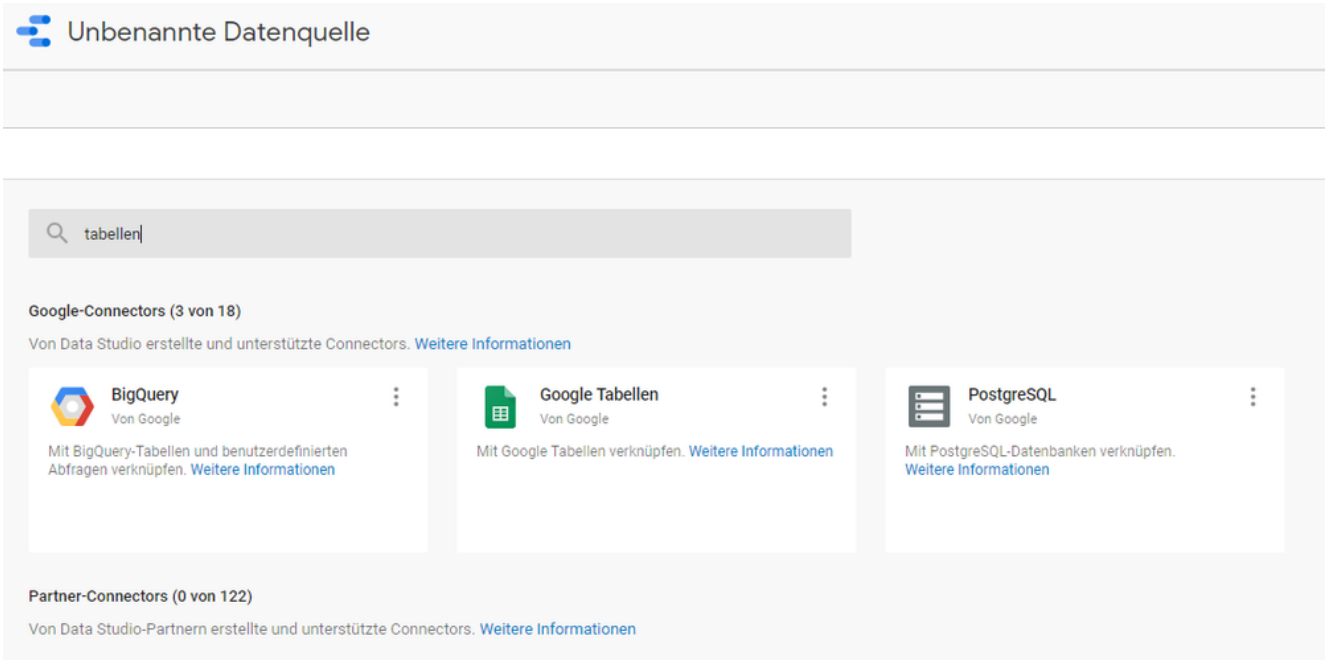


Abb. 9: Auswahl von Google-Tabellen als Data-Studio-Connector

Innerhalb des Berichtes besteht die Möglichkeit, mit mehreren Seiten zu arbeiten und damit bspw. die Sheets-Integration als Detailbericht einer großen Analyse zu nutzen. Weiterhin können einzelne Informationen/Spalten aus Sheets ausgewählt und in diversen Visualisierungsmöglichkeiten auf dem Bericht verortet werden.

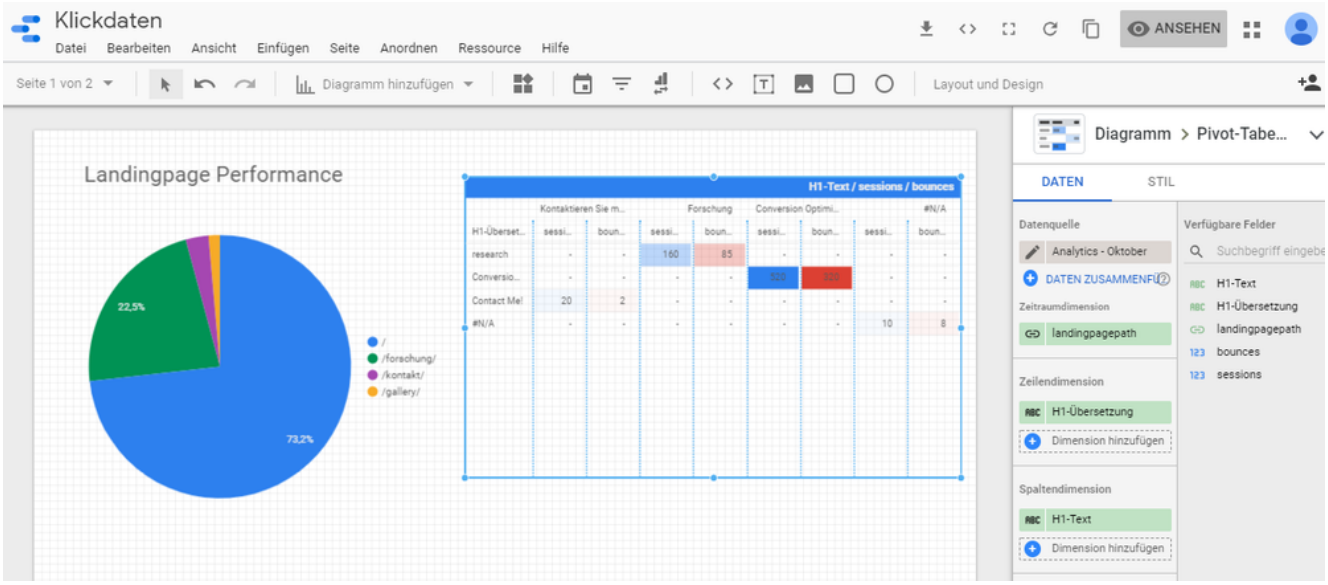


Abb. 10: Berichtsaufbau mit Sheets-Integration

Fazit

Zusammenfassend lässt sich sagen, dass R, Google Sheets, Google Data Studio etc. für sich hervorragende Möglichkeiten bieten. Die wahre Stärke zeigt sich, wie fast immer, in der Kombination verschiedener Faktoren/Quellen. Daten aus diversen Quellen (schnell) zusammenzufassen, aus unterschiedlichen Blickwinkeln zu betrachten und unkompliziert einem definierten Empfängerkreis zur Verfügung zu stellen, sind nur punktuelle Vorteile.

In diesem Sinne: Verbinden Sie Ihre Informationen, kombinieren Sie Daten – ggf. trügt ja der erste Anschein.

R4SEO: Automatisierte Reports mit R generieren (Teil 1)

R-Leuchtungen! Teil 9

R-Leuchtungen! Teil 9: Title-Geddon – war da was bei Ihnen? – websiteboosting.com

In den Ausgaben bis 54 bis 58 der Website Boosting konnten Sie in der Serie „R4SEO“ von Patrick Lürwer nachvollziehen, wie man die kostenlose Software R verwendet, was sie leistet und wie man sie nutzbringend für die eigene Arbeit für SEO bzw. die Aufklärung im Online-Marketing einsetzen kann. R…
In der letzten Ausgabe der Website Boosting gab Tobias Aubele einen sehr guten Einblick in die Potenziale von R, um auf einfache und reproduzierbare Weise Daten über die Google-Analytics-API abzufragen. Diese neue Artikelserie von Patrick Lürwer baut darauf auf und zeigt, wie mittels R Reports erstellt werden können, die sich zu einem bestimmten Zeitpunkt automatisch aktualisieren und per Mail verschicken. Im ersten Teil geht es nun zunächst um das Abfragen verschiedener APIs für die Datenbeschaffung. In den folgenden Teilen werden die Daten aggregiert, visualisiert und in einem Report zusammengefasst.

Tipp

Um Ihnen das Abtippen des Codes zu ersparen, können Sie sich das Skript zu diesem Artikel unter folgendem Link auf GitHub herunterladen. Dennoch empfehle ich Ihnen, den Code selbst zu schreiben, um ein Gefühl für die Syntax zu bekommen. einfach.st/r4seocode

Das Google Data Studio (GDS) bietet eine einfache Möglichkeit, Daten aus dem Google-Kosmos – bspw. aus Google Analytics (GA) und der Google Search Console (GSC) – abzufragen, zu visualisieren und als Reports aufzubereiten. Problematisch wird es allerdings, wenn andere Datenquellen angebunden werden sollen. Denkbar sind hier APIs anderer Anbieter wie Sistrix. In solchen Fällen muss der „Umweg“ über Google Spreadsheets und Google Apps Scripts gegangen werden, um die Abfragen selbst zu programmieren. Dies ist durchaus ein valides

Vorgehen, das jedoch schnell an seine Grenzen stoßen kann. Denn meist sind Reports nur der erste Schritt. Sobald in diesen Auffälligkeiten erkennbar werden, schließt eine Analyse an, die eine umfassende Beschäftigung mit den zugrunde liegenden Daten erfordert. Derartige Analysen verlangen zumeist komplexe Datentransformationen und -aggregationen, für die Dashboard-Softwares, wie das GSA, mit ihren begrenzten Filtern- und Manipulationsoptionen naturgemäß nicht ausgelegt sind. Ganz abgesehen davon, dass der direkte Zugriff auf die Daten zur Weiterverarbeitung häufig gar nicht gegeben ist. Von großem Vorteil ist es daher, wenn Report- und Analysesoftware in einer Anwendung vereint sind. Diese Anforderungen kann R insofern erfüllen, als die Programmiersprache durch Packages so erweitert werden kann, dass mit ihr sowohl sehr individualisierte Reports erstellt als auch umfangreiche Analysen durchgeführt werden können.

Da Datenanalysen eine sehr komplexe Thematik sind, soll in diesem Artikel jedoch zunächst das Potenzial von R zur Report-Generierung vorgestellt werden. Denn im Grunde sind beide Aufgaben bis zu einem gewissen Grade sehr ähnlich. Zunächst werden Daten von verschiedenen Quellen abgefragt, dann aufbereitet und schließlich visualisiert. Bei Analysen ist dieser Prozess jedoch iterativ, ohne dass im Vorhinein klar ist, wie das Ergebnis resp. die Erkenntnis genau aussieht. Bei der Generierung von Reports hingegen ist der Ablauf des Datenprozesses vorgegeben. Sie sind daher kleine, standardisierte Analysen, die an einem bestimmten Punkt der Visualisierung aufhören. Das Ergebnisdokument ist somit immer gleich und jederzeit reproduzierbar. Daher eignen sie sich sehr gut, um exemplarisch die Möglichkeiten von R aufzuzeigen, und sind dabei aber gleichzeitig so vereinheitlicht, um durch den Leser selbst nachgebaut zu werden. Ich möchte Sie daher gerne dazu auffordern, das folgende Skript selbst in RStudio umzusetzen.

Hinweis

Der Autor ist sich bewusst, dass gerade für Einsteiger der Anfang etwas holprig sein kann. Aber auch bei erfahreneren Lesern können durchaus Fragen aufkommen. Um hier Hilfestellung zu bieten, die Antworten der Community zugänglich zu machen und den Austausch anzuregen, möchte er Sie ermutigen, in der extra hierfür gegründeten Facebook-Gruppe nachzufragen. Es gibt dabei keine „dummen“ Fragen. Vielmehr hätte der Autor sich in seiner Anfangszeit selbst ein solches Forum mit dem Fokus auf analytische Programmierung und Online-Marketing sehr gewünscht. www.facebook.com/groups/ompyr/

Aber genug der langen Vorrede! In diesem Teil der Serie ist das Ziel, die Sessions aus GA sowie die Performance-Daten aus der GSC abzufragen, um die Packages für diese APIs besser kennenzulernen. Darüber hinaus werden Sie am Beispiel von Sistrix eine eigene API-Abfrage schreiben, um den Sichtbarkeitsindex für eine Domain zu ermitteln. Als Beispiel-Domain dient hier die Website dein-trueffel.de, ein Studentenprojekt an der Hochschule Darmstadt im Modul „Fortgeschrittene Suchmaschinenoptimierung“, das durch Jens Fauldrath und Stefan Keil betreut wird. Voraussetzung zum Nachvollziehen des Skripts sind eine aktuelle R-Installation (www.r-project.org) und RStudio (www.rstudio.com).

Anlegen eines neuen Projekts in RStudio

Zunächst wird ein neues Projekt angelegt (File → New Project … → New Directory → New Project; Abb. 1). Benennung und Speicherort können Sie frei wählen. Der Einfachheit halber wird für diesen Beitrag das Projekt „report“ (Directory name) unter C:\Users\{USER}\Documents\ (im Eingabedialog durch ~ automatisch abgekürzt; Abb. 2) erzeugt.

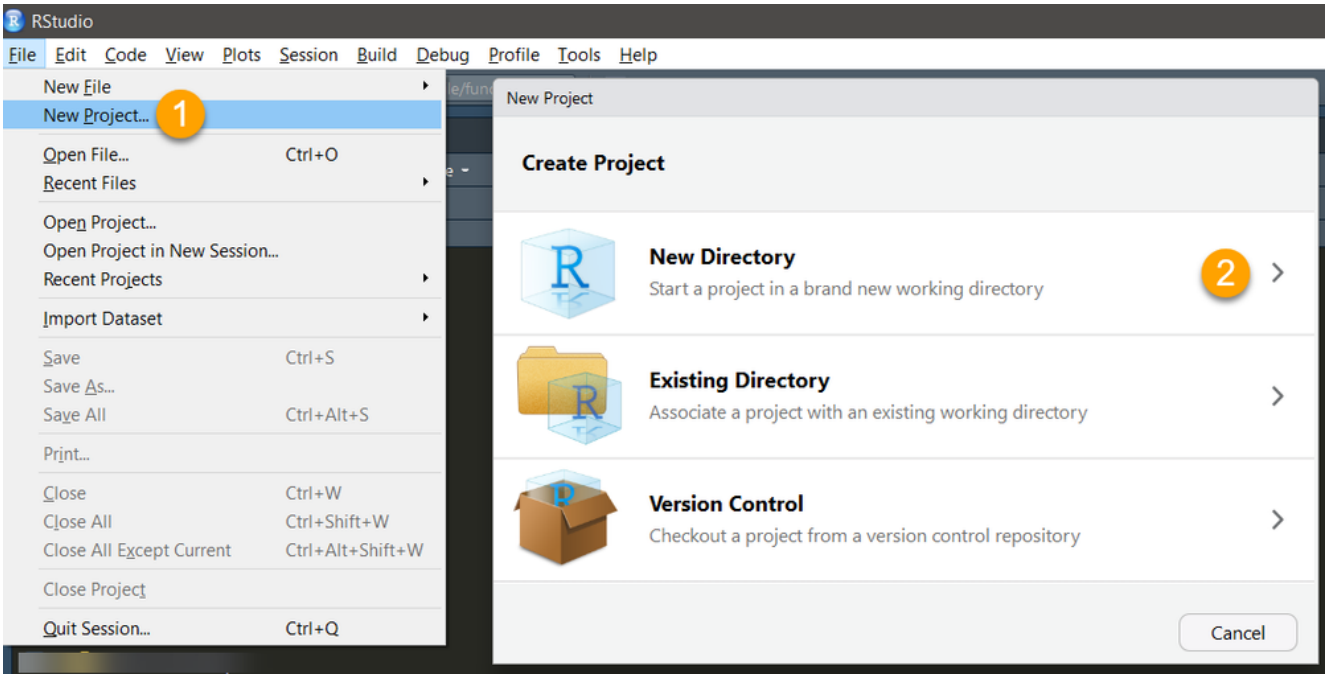


Abbildung 1: Anlegen eines neuen Projekts

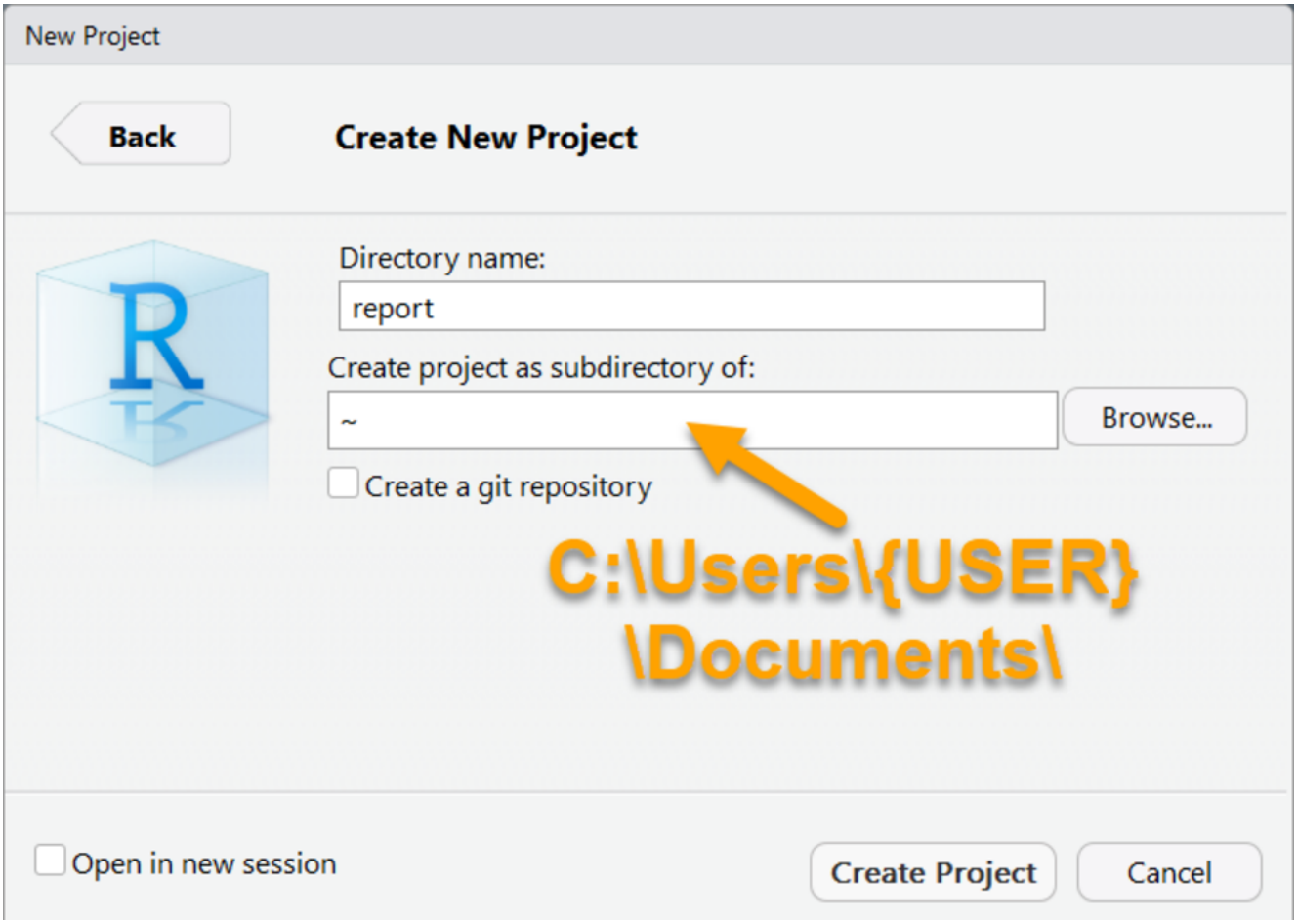


Abbildung 2: Speicherort und Benennung des Projekts

Das Projekt ist zunächst leer, im File-Explorer finden Sie nur eine RStudio-Projektdatei (report.Rproj), die ignoriert werden kann. Daher wird als erstes über File → New File → R Script (oder den darunter befindlichen Button) eine neue Datei

erzeugt und unter der Benennung api_calls.R gespeichert.

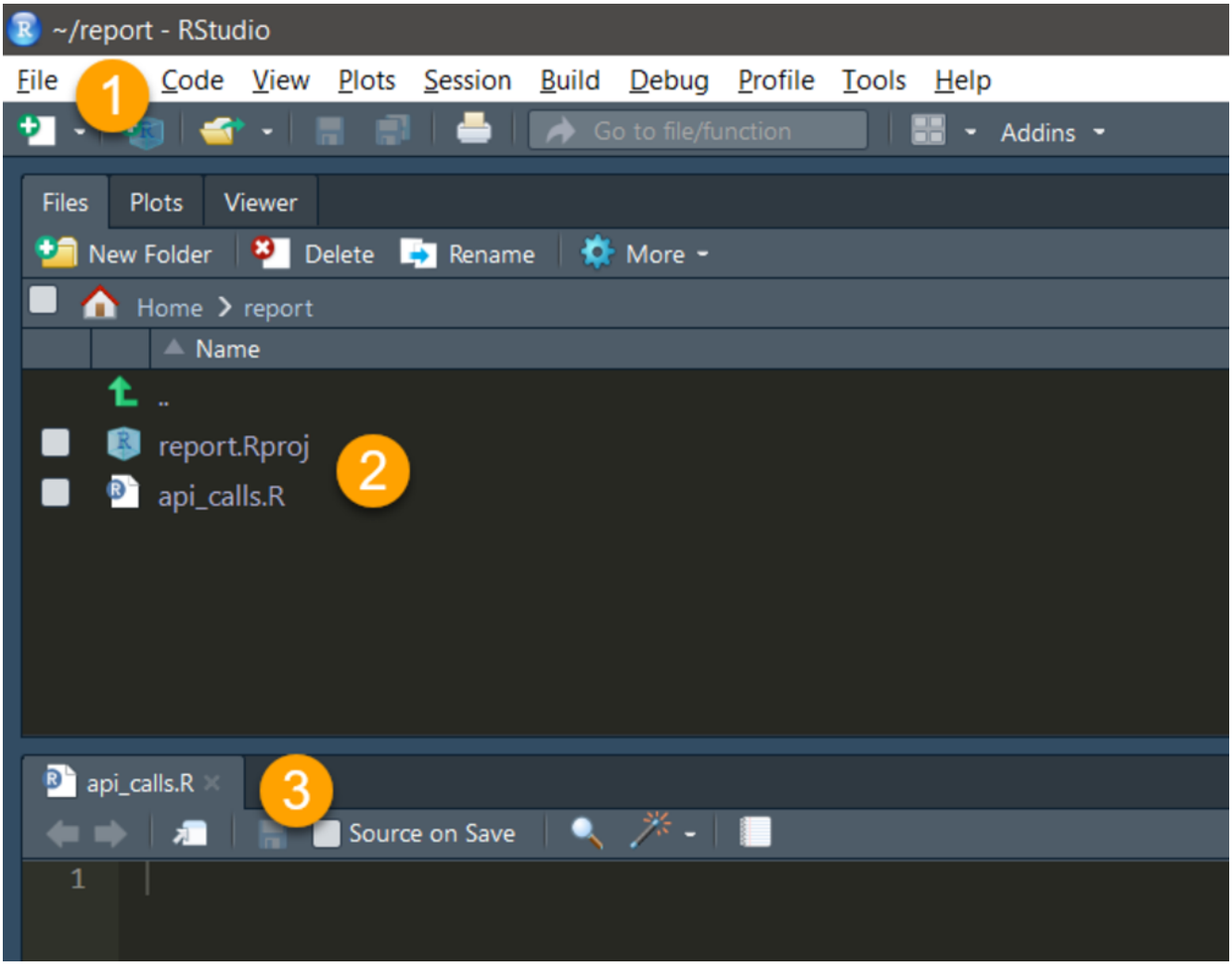


Abbildung 3: (1) Neues Skript erstellen, (2) Projekt-Datei und Skript im File-Explorer, (3) Skript-Editor zeigt die erstellte Datei an

Installation und Laden der benötigten Packages

In der soeben erstellen api_calls.R-Datei werden nun die benötigten Packages installiert bzw. – falls bereits vorhanden – aktualisiert und geladen (Abb. 4; 1). Die Google-Packages kennen Sie bereits aus dem Beitrag von Tobias Aubele. Das Package lubridate erweitert den Funktionsumfang von R für die komfortable Arbeit mit Datums- und Zeitwerten, jsonlite dient zum Abfragen und Verarbeiten von JSON-Dateien. Sobald Funktionen aus diesen Packages verwendet werden, wird explizit

darauf hingewiesen. Die vorangestellten Rauten kommentieren den Code aus. Beim ersten Ausführen müssen Sie diese also entfernen. Markieren Sie dann den gesamten Code mit der Maus und führen Sie ihn durch Drücken der STRG- und ENTER-Tasten aus.

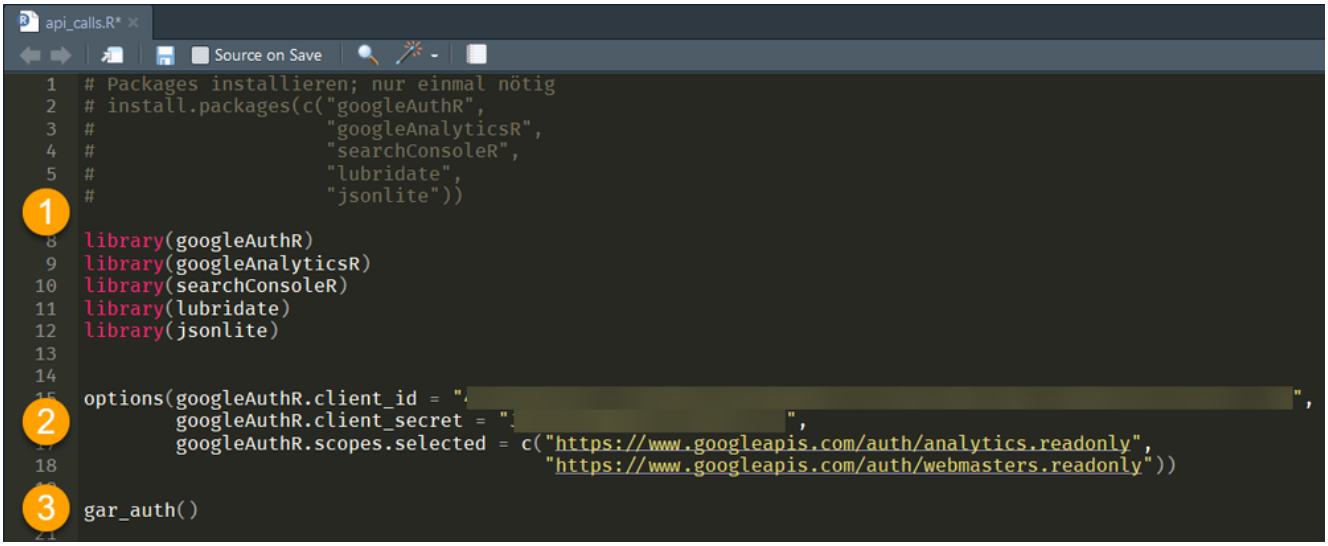


Abbildung 4: Installieren bzw. Aktualisieren und Laden der benötigten Packages

In der anschließenden options()-Funktion definieren Sie Ihre Google-Client-ID sowie Ihr -Secret (2). Diese können Sie sich unter console.developers.google.com generieren. Das Argument scopes definiert die Berechtigungen für die GA- bzw. GSC-API. gar_auth() dient schließlich der Authentifizierung (3). Führen Sie den Code das erste Mal aus, öffnet sich Ihr Browser, in dem Sie den Zugriff des Skripts bestätigen müssen. Ist dies erfolgt, finden Sie in Ihrem File-Explorer eine Authentifizierungsdatei (sc.oauth). Bei einem erneuten Ausführen des Skripts verwendet gar_auth() diese Datei, sodass Sie nicht erneut eine manuelle Authentifizierung durchführen müssen – praktisch, denn zukünftig soll das Skript ja automatisch laufen.

Konfiguration wichtiger Variablen

Nun folgt ein Code-Abschnitt, in dem einige Variablen definiert werden (Abb. 5), die an verschiedenen Stellen des Skripts zum Einsatz kommen. Dass die Variablen hier

großgeschrieben werden, ist eine Konvention des Autors, die anzeigt, dass die Variablen erst zu einem späteren Zeitpunkt genutzt werden – grundsätzlich könnten Sie die Variablen auch kleinschreiben. Dies ist somit die Stelle im Skript, an der Sie Ihre eigenen Daten eintragen müssen.

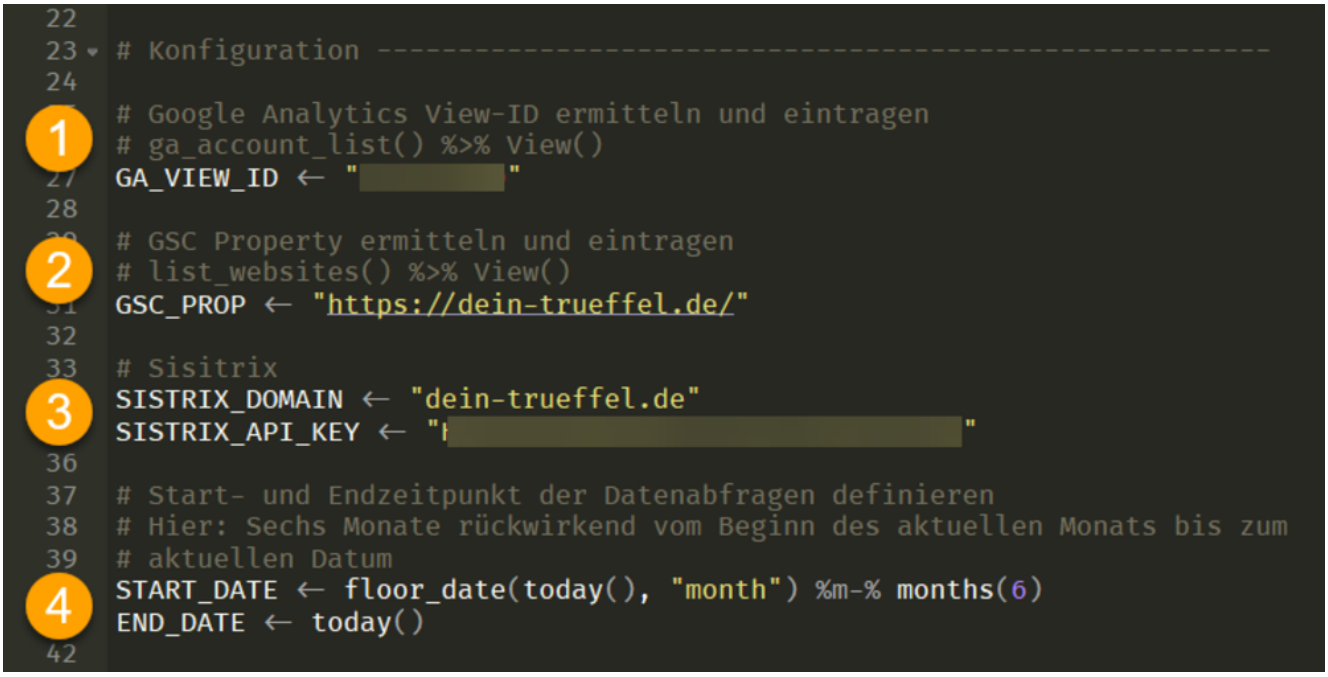


Abbildung 5: Definition globaler Variablen, für die die Daten von den APIs abgefragt werden sollen
Im Detail:

(1) Um die passende View-ID Ihrer GA-Datensicht zu ermitteln, können Sie die auskommentierte Funktion ga_account_list() %>% View() verwenden. Der Pipe-Operator %>% übergibt den von der API zurückgegebenen DataFrame mit den Account-Details an die Funktion View(), welche die Tabelle in einem Explorer statt auf der Console öffnet (Abb. 6). Falls der letzte Satz nur ein böhmisches Dorf für Sie war, seien Sie beruhigt, eine ausführlichere Erklärung folgt im nächsten Artikel. Wichtig ist erst einmal nur, dass Sie Ihre View-ID definieren. Entnehmen Sie dazu die gewünschte View-ID aus der Spalte viewId und tragen Sie sie bei der Variable GA_VIEW_ID ein.

(2) Analog dazu verfahren Sie mit der siteUrl für die Variable GSC_PROP.

(3) Die Sistrix-Daten sollen später auf Domain-Ebene abgefragt werden, entsprechend tragen Sie bei SISTRIX_DOMAIN diese ein. Ihren Sistrix-API-Key für die Variable SISTRIX_APY_KEY finden Sie unter app.sistrix.com/account/api.

(4) START_DATE und END_DATE definieren den abzufragenden Berichtszeitraum. In der aktuellen Konfiguration umfasst dieser sechs Monate rückwirkend vom Beginn des aktuellen Monats bis zum heutigen Datum. floor_date(today(), „month“) %m-% months(6) bedient sich dreier Funktionen aus dem Package lubridate. today() gibt das heutige Datum zurück („2018-01-11“). floor_date(„2018-01-11“), „month“) rundet dann zum Beginn des aktuellen Monats („2018-01-01“). %m-% months(6) subtrahiert von diesem Datum sechs Monate („2018-07-01“). %m-% ist ein besonderer Operator aus dem Package, der das Subtrahieren von Monaten unter Berücksichtigung unterschiedlicher Tageszahlen im Monat und Schaltjahren unterstützt. Sie landen mit dieser Funktion also immer am ersten eines Monats.

Die definierten Variablen können Sie nun im Environment sehen (Abb. 7).

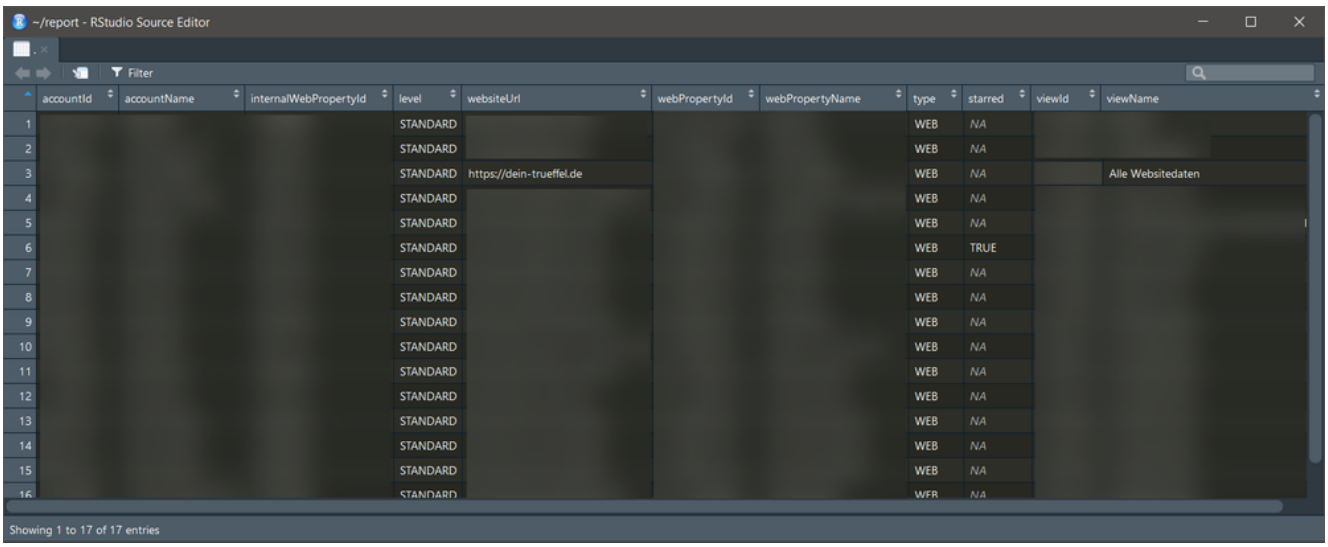


Abbildung 6: Data-Explorer mit den GA-Accounts

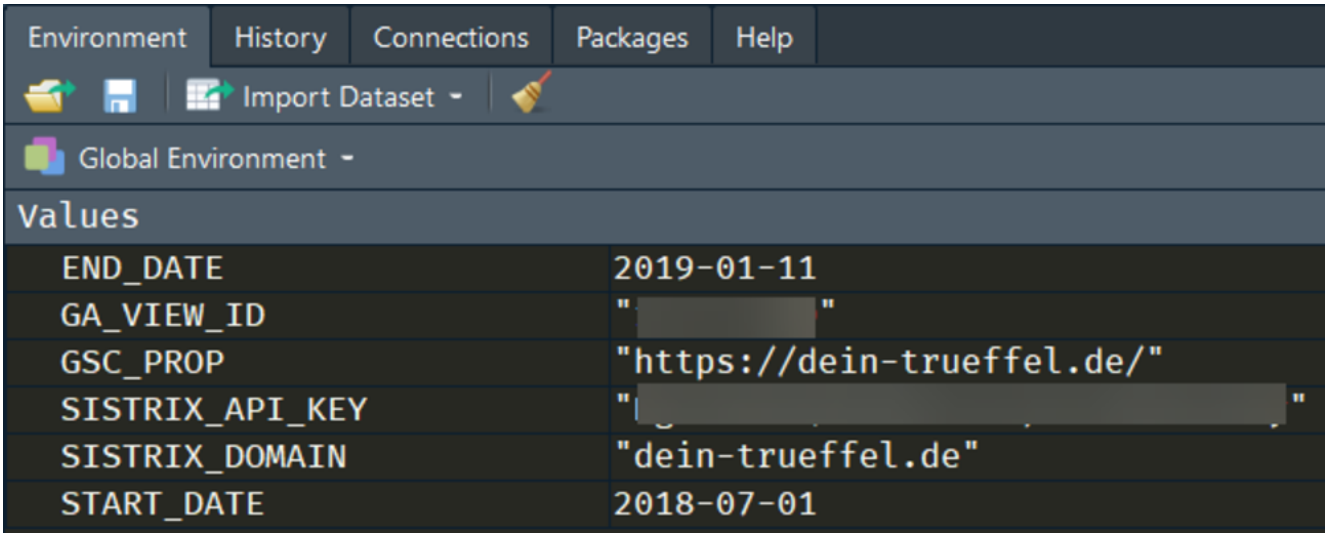


Abbildung 7: Environment oder Variable-Explorer listet alle zugewiesenen Variablen

GA-Sessions abfragen

Wenn die Konfiguration abgeschlossen ist, kann mit der ersten API-Abfrage begonnen werden. Als Erstes sollen die Sessions, die aus den Channels Organic Search resp. Direct resultieren, aus GA abgerufen werden (Abb. 8). Um die Abfrage auf diese Dimensionen einzuschränken, werden mittels dim_filter() zwei Filter-Komponenten konfiguriert und in der Variablen channel_filter_organic bzw. channel_filter_direct gespeichert (1). Diese werden an eine weitere Funktion filter_clause_ga4() gegeben, welche intern die Filter in eine für die API verständliche Syntax übersetzt (2). Die Funktion google_analytics() führt schließlich die eigentliche Abfrage aus (3). Hier sehen Sie auch die Variablen für die GA-Datensicht sowie das Start- und Enddatum, die Sie zuvor konfiguriert haben. Führen Sie den Code aus, erscheint in Ihrem Environment die Variable ga_sessions (Abb. 9; 1), die Sie mit der Maus anklicken können und die einen DataFrame mit der API-Antwort enthält (2).

```r
# Google Analytics ----------------------------------------------------

# Organic Search / Direct Sessions abfragen

channel_filter_organic <- dim_filter(dimension = "channelGrouping",
                                     operator = "EXACT",
                                     expressions = "Organic Search")

channel_filter_direct <- dim_filter(dimension = "channelGrouping",
                                    operator = "EXACT",
                                    expressions = "Direct")

filter_clause <- filter_clause_ga4(list(channel_filter_organic,
                                        channel_filter_direct),
                                   operator = "OR")

ga_sessions <- google_analytics(viewId = GA_VIEW_ID,
                                date_range = c(START_DATE, END_DATE),
                                metrics = c("sessions"),
                                dimensions = c("date", "channelGrouping"),
                                dim_filters =  filter_clause,
                                anti_sample = TRUE)
```

Abbildung 8: API-Abfrage der GA-Sessions

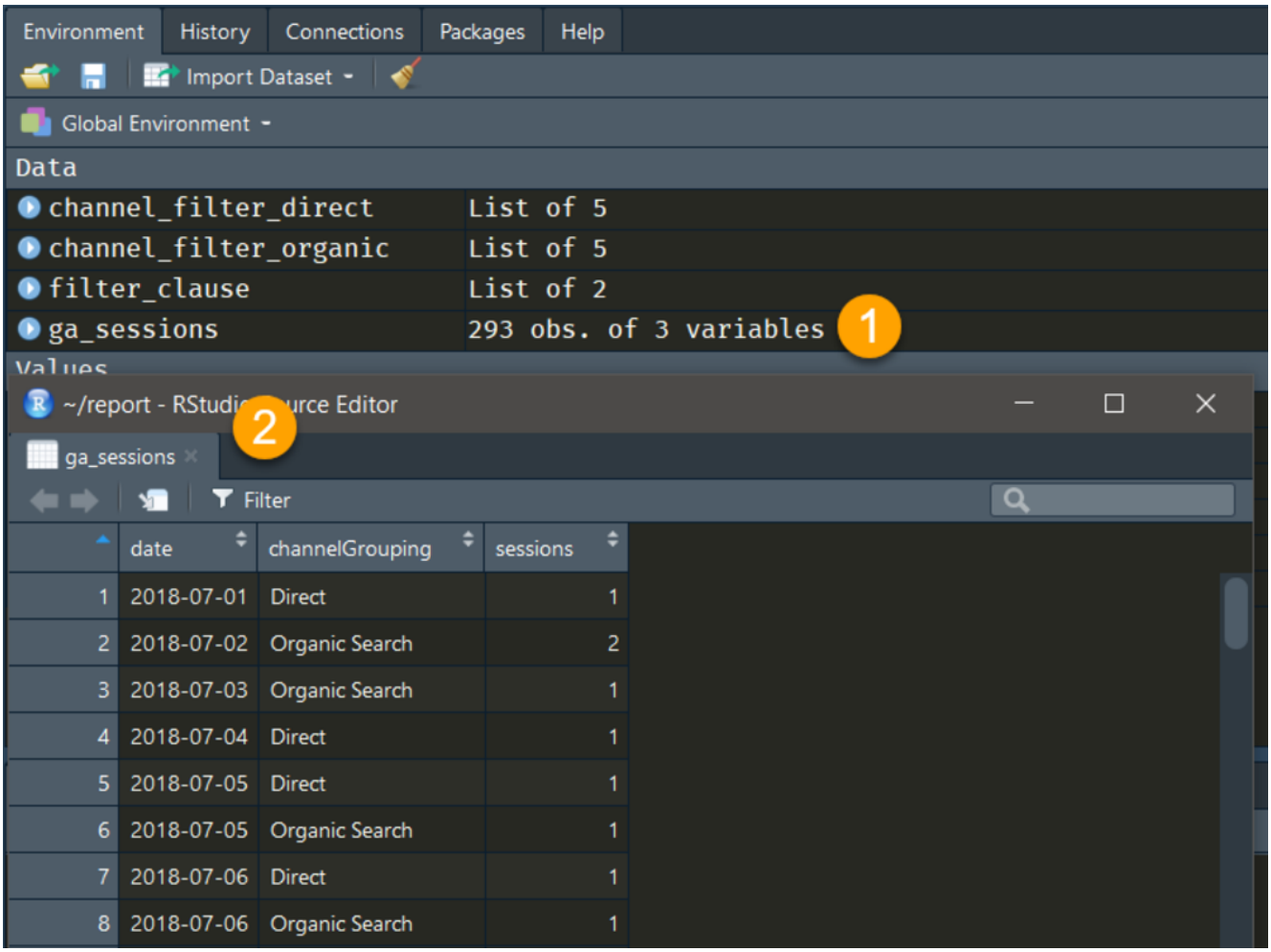


Abbildung 9: DataFrame mit den GA-Sessions

GSC-Performance-Daten abfragen

Ähnlich funktioniert die anschließende Abfrage der GSC-API (Abb. 10). Von dieser werden verschieden Dimensions-Kombinationen abgefragt. (1) Die erste Abfrage ruft die Clicks, Impressions, CTR und Position für den searchType „web“ auf Tagesbasis ab. Auch hier sehen Sie die Verwendung der anfangs definierten Variablen. (2) Dann erfolgt die Abfrage der Metriken für die Kombination aus Datum und Suchanfrage sowie schließlich (3) für die Kombination aus Datum und URL. Auch hier finden Sie nach dem Ausführen der API-Calls die Antwort-Tabellen in Ihrem Environment.

```
67
68  # GSC ----------------------------------------------------------------
69
70  # GSA: Date
71
1   gsa_dim_date ← search_analytics(siteURL = GSC_PROP,
                                    startDate = START_DATE,
74                                  endDate = END_DATE,
75                                  searchType = "web",
76                                  dimensions = "date")
77
78  # GSA: Date ~ Query
79
2   gsa_dim_date_query ← search_analytics(siteURL = GSC_PROP,
                                          startDate = START_DATE,
82                                        endDate = END_DATE,
83                                        searchType = "web",
84                                        dimensions = c("date", "query"))
85
86  # GSA: Date ~ Page
87
3   gsa_dim_date_page ← search_analytics(siteURL = GSC_PROP,
                                         startDate = START_DATE,
90                                       endDate = END_DATE,
91                                       searchType = "web",
92                                       dimensions = c("date", "page"))
93
```

Abbildung 10: API-Abfrage der GSC-Metriken für verschiedene Dimensionskombinationen

Sistrix-Sichtbarkeitsindex abfragen

Zu guter Letzt müssen Sie noch selbst eine API-Abfrage schreiben, denn es gibt natürlich nicht für alle Datenquellen fertige Packages. Sistrix ist so ein Fall, der aber aufgrund

der sehr einfachen Abfrage-Syntax ohne großen Aufwand umgesetzt werden kann. Die Abfrage ist nämlich nur eine URL, deren Query-Parameter die gewünschten Abfragewerte aufnehmen. Wird die URL aufgerufen, erhalten Sie als Antwort eine JSON-Datei. Diese URL bauen Sie mit der Funktion paste0() zusammen (Abb. 11; 1). Konkret verbindet die Funktion die in Klammern angegebenen Argumente. Diese bestehen aus den Bestandteilen der Basis-URL für die Abfrage des historischen Sichtbarkeitsindexes (SI), in die die zuvor definierten Variablen für Ihren API-Key sowie die abzufragende Domain eingefügt werden.

Anschließen fragen Sie die zusammengebauten URLs gegen die API an (2). Dazu verwenden Sie die Funktion fromJSON() aus dem Package jsonlite. Die Funktion führt einen GET-Request aus, nimmt die JSON-Antwort der API entgegen und überführt sie automatisch in eine Liste – ein R-Datentyp zum Speichern hierarchischer Daten. Die Antwort wird zunächst in die jeweiligen Variablen geschrieben. Klicken Sie eine der beiden Variablen im Environment an, sehen Sie, dass das JSON in eine Listen-Struktur überführt wurde (Abb. 12). Hier sehen Sie auch schon den „Weg“, den Sie innerhalb der Liste gehen müssen, um an die SI-Daten heranzukommen. Denn neben diesen erhalten Sie noch weitere Meta-Informationen, wie bspw. die verbrauchten Credits, zurück. Um die Tabelle mit den SI-Daten zu extrahieren, müssen Sie sich an den übergeordneten Listenpunkten entlanghangeln. Dies macht der Ausdruck api_response_desktop$answer$sichtbarkeitsindex[[1]] für den Desktop-SI und überführt den DataFrame in die Variable si_desktop (Abb. 11; 3).

```
# Sistrix ----------------------------------------------------------

# API-URLs zusammenbauen
si_url_desktop ← paste0("https://api.sistrix.com/domain.sichtbarkeitsindex?history=true&format=json&api_key=",
                        SISTRIX_API_KEY,
                        "&domain=",
                        SISTRIX_DOMAIN)

si_url_mobile ← paste0("https://api.sistrix.com/domain.sichtbarkeitsindex?history=true&format=json&api_key=",
                       SISTRIX_API_KEY,
                       "&domain=",
                       SISTRIX_DOMAIN,
                       "&mobile=true")

# API abfragen
api_response_desktop ← fromJSON(si_url_desktop)
api_response_mobile ← fromJSON(si_url_mobile)

# JSON-Antwort in DataFrame überführen
si_desktop ← api_response_desktop$answer$sichtbarkeitsindex[[1]]
si_mobile ← api_response_mobile$answer$sichtbarkeitsindex[[1]]
```

Abbildung 11: Konstruktion der Sistrix-API-Abfrage

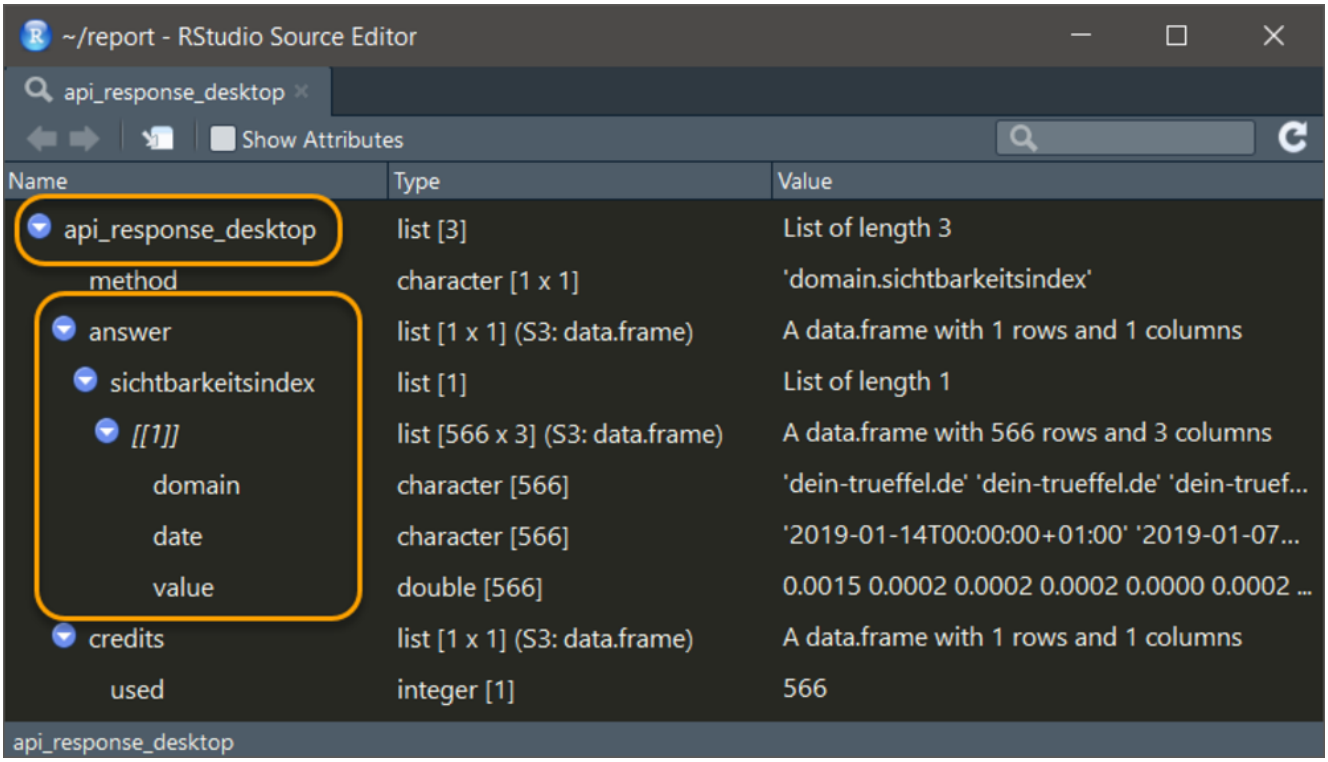


Abbildung 12: Liste mit der Antwort der Sistrix-API

Fazit und Ausblick

Damit haben Sie das Ziel dieses Beitrags erreicht. Sie haben ein reproduzierbares Skript geschrieben, dass Sie jederzeit erneut ausführen können. Der Berichtszeitraum der abzufragenden Daten passt sich bei jeder erneuten Ausführung automatisch an. Als Resultat liegen Ihnen die GA- , GSC- und Sistrix-Daten in tabellarischer Form vor. Im nächsten Artikel dieser Reihe wird es dann um die Aufbereitung dieser Daten gehen. Aktuell liegen die Daten auf Tagesbasis vor. Für den

Report sollen bspw. die GSC-Daten auf Query- resp. URL-Basis aggregiert werden, um somit Tabellen der Top-10-Suchphrasen bzw. URLs der Vorwoche zu erstellen.

R4SEO: Automatisierte Reports mit R generieren (Teil 2)

R4SEO: Automatisierte Reports mit R

generieren (Teil 2) – websiteboosting.com

Im ersten Teil dieser Artikelserie in der letzten Ausgabe hat Ihnen Patrick Lürwer gezeigt, wie Sie mit R die Google-Analytics- und Search-Console- sowie SISTRIX-API abfragen können. Im vorliegenden Teil werden Sie lernen, die Daten so aufzubereiten, dass sie zur Darstellung im Report geeignet…
Im ersten Teil dieser Artikelserie in der letzten Ausgabe hat Ihnen Patrick Lürwer gezeigt, wie Sie mit R die Google-Analytics- und Search-Console- sowie SISTRIX-API abfragen können. Im vorliegenden Teil werden Sie lernen, die Daten so aufzubereiten, dass sie zur Darstellung im Report geeignet sind. Dazu werden Sie fehlende Datumspunkte in den DataFrames ergänzen. Sie werden den gleitenden Mittelwert für verschieden Metriken berechnen, um starke Schwankungen der Tagesdaten zu glätten. Außerdem erfahren Sie, wie Sie DataFrames transponieren, um sie von einem weiten in ein langes Format zu bringen.

Im letzten Artikel haben Sie Daten aus drei unterschiedlichen Quellen abgefragt, sodass Ihnen diese nun in tabellarischer Form (DataFrames) in R vorliegen. Aber wie das so häufig ist, haben die APIs die Daten nicht so geliefert, wie sie für die Report-Erstellung benötigt werden. Beispielsweise liefert die Google-Search-Console-API (GSC) nur Datumspunkte zurück, an denen auch tatsächlich Impressions bzw. Clicks stattgefunden haben. Es fehlen folglich die Tage ohne Impressions. Außerdem liegen die Daten auf Tagesbasis vor und sind demzufolge unter Umständen sehr volatil. Sie wollen sie daher glätten, um Trends besser erkennen zu können. Es ist somit unabdingbar, dass Sie die Daten „aufräumen“ und in eine entsprechende Form bringen, um sie für die Visualisierung verwenden zu können. Im Englischen spricht man in dem Fall von *tidying* – daher auch die Benennung des Packages *tidyverse*, das Sie verwenden werden. Viele der darin enthaltenen Funktionen dienen allein dazu, Daten zunächst in die richtige Form zu bringen, um mit ihnen arbeiten zu können.

Neues Skript erstellen sowie Daten und Packages laden

Bevor Sie mit dem Aufräumen beginnen, generieren Sie in Ihrem Projekt, das Sie im Zuge des letzten Artikels angelegt haben, ein neues R-Skript mit der Benennung *data_tidying.R* und kopieren Sie den gesamten Inhalt des Skripts *api_calls.R* hinein. Führen Sie das Skript aus, damit die aktuellen Daten von den APIs abgefragt werden und zur Weiterverarbeitung in DataFrames in Ihrem Environment vorliegen (Abb. 1).

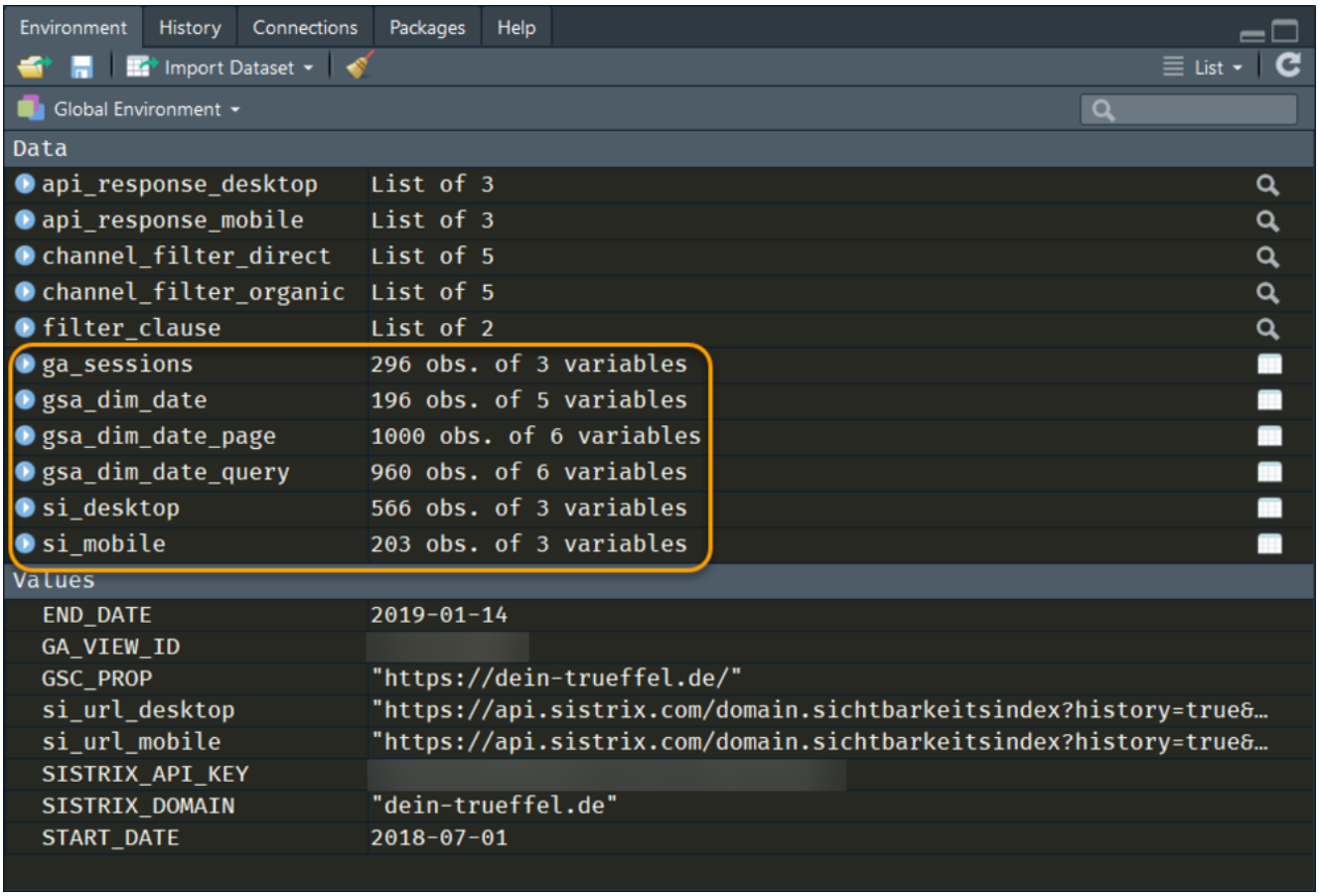


Abbildung 1: Environment mit geladenen API-Daten

Im Anschluss installieren und laden Sie wieder einige Packages, die für die nachfolgenden Datentransformationen benötigt werden (Abb. 2). *tidyverse* (https://www.tidyverse.org/) ist eine sehr mächtige Sammlung von Packages für die Datenmanipulation. In diesem Package ist auch *ggplot* enthalten, welches Sie schon aus dem Artikel von Tobias Aubele kennen und das zum Plotten von Daten dient.

Darüber hinaus erweitert es die R-Syntax um den Pipe-Operator (%>%), den Sie im vorherigen Artikel kurz verwendet haben. *zoo* und *padr* dienen dem Arbeiten mit Zeitreihen.

Tipp

Eine sehr anschauliche Einleitung in die Syntax und gängige Funktionen des tidyverse finden Sie unter folgendem Link. Das dort beschriebene Package dplyr ist ein Bestandteil des tidyverse und dient der Datenmanipulation, wie Sie sie im Folgenden selbst durchführen werden: speakerdeck.com/omaymas/data-manipulation-with-dplyr-first-steps

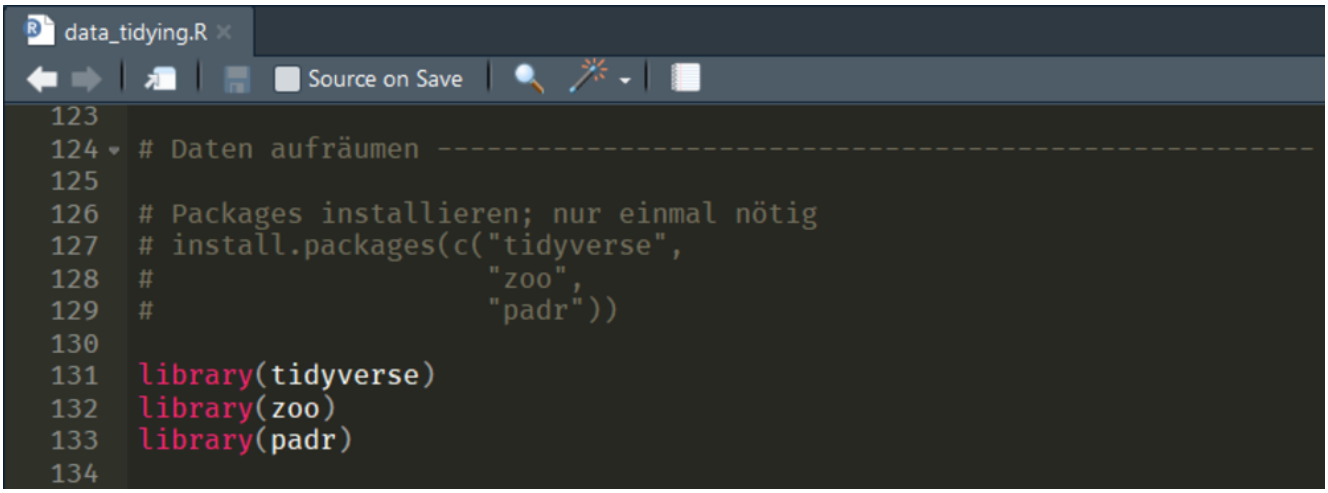


Abbildung 2: Installieren bzw. Aktualisieren und Laden der benötigten Packages

Google-Search-Console-Daten vervollständigen

Wie bereits angesprochen, gibt die GSC-API nur Datumspunkte zurück, an denen auch tatsächlich mindestens eine Impression stattgefunden hat. Werden die Daten nur mit der Dimension *date* abgefragt, wie Sie es für die Variable *gsa_dim_date* getan haben, liegen meistens für alle Tage Daten vor, denn hier wird die gesamte Website betrachtet. Wird die Dimension *date* jedoch um die *query* erweitert, ist dies nicht mehr der Fall, denn

nicht jedes Keyword wird an jedem Datum des Berichtszeitraums eine Impression generiert haben. Dies sehen Sie in Abbildung 3, in der die Anzahl der rankenden Keywords gezählt wurde. Um die Tabelle selbst zu reproduzieren, können Sie den Befehl, den Sie im Screenshot sehen, in der Console ausführen. Dadurch ist er kein Bestandteil des Skripts, sondern dient nur dazu, die Daten on-the-fly anzuzeigen. 2018-07-02 hat nur ein Keyword Impressions generiert, 2018-07-05 zwei. An den Tagen dazwischen gab es keine Rankings, sodass keine Datumspunkte von der API zurückgegeben wurden.

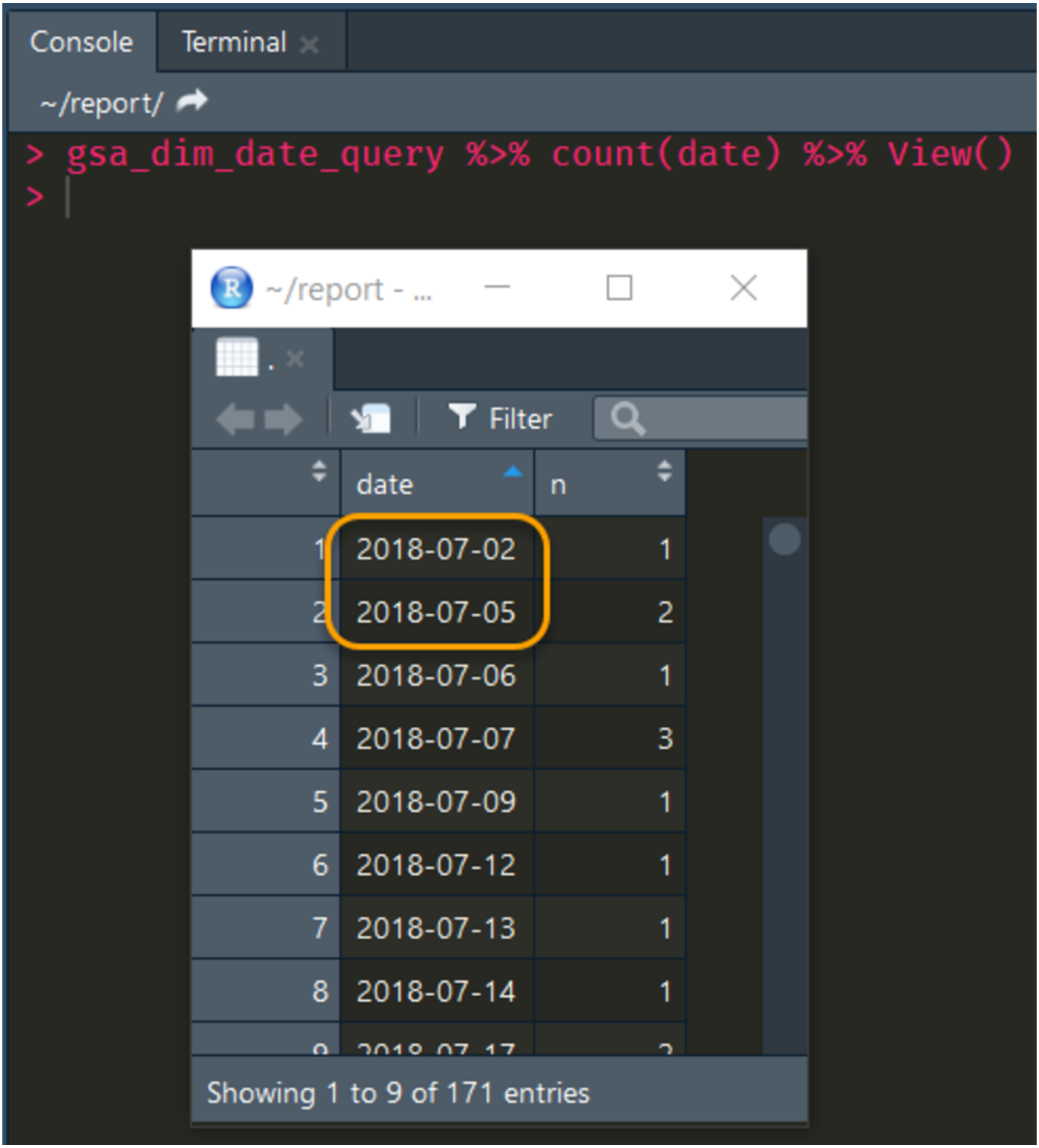


Abbildung 3: Zählen der Queries je Tag; fehlende Datumswerte an Tagen ohne Impressions

Wie geschrieben werden fehlende Datumspunkte bei Abfragen, die allein aus der Dimension *date* bestehen, recht selten vorkommen, da dies bedeuten würde, dass keine einzige Seite an diesem Tag eine Impression generiert hat. Sie müssen sich aber stets bewusst darüber sein, dass dieses Verhalten auftreten kann. Und dann möchten Sie dies auch bestimmt in Ihren Diagrammen sehen. Denn wenn an einem Tag keine Seite ein

Ranking aufweisen kann, liegt definitiv etwas im Argen. Solche fehlenden Werte müssen Sie also explizit machen, denn standardmäßig werden beim Plotting Werte für fehlende Datumspunkte interpoliert. In Abbildung 4 sehen Sie einen exemplarischen Verlauf von Impressions je Tag im Monat, wobei die Anzahl der Impressions der Einfachheit halber konstant ist. Für die Datumspunkte 5, 8, 12 / 13, 24 und 30 liegen keine Werte vor. Die blaue Linie (1) wurde ohne ergänzte Datumspunkte geplottet. Hier sehen Sie, dass die Linie zwischen den bestehenden Datumspunkten einfach weitergezogen wurde. Der fehlende Datumspunkt 5 wird durch Interpolation (vereinfacht: dem Mittelwert der beiden bekannten umliegenden Werte) berechnet. Bei der roten Linie (2) wurden die fehlenden Datumspunkte ergänzt, allerdings keine Ersatzwerte eingetragen. Dadurch bricht die Linie an diesen Punkten ab. Die fehlenden Werte liegen nun explizit in den Daten vor. Das Zielbild ist aber die grüne Linie (3). Bei dieser wurden die Datumspunkte ergänzt und die fehlenden Werte durch 0 aufgefüllt. Hier fällt somit bei der Betrachtung des Graphen sofort ins Auge, dass die Impressions auf 0 abgestürzt sind.

Das Auffüllen ist im Übrigen nicht nur beim Plotting wichtig, sondern auch bei der Aggregation der Daten auf einer höheren Zeitebene. Soll beispielsweise der Durchschnitt der Impressions auf Monatsbasis berechnet werden, verfälschen fehlende Werte das Ergebnis. Angenommen, jeder Datumspunkt im Diagramm zeigt 10 Impressions an, wäre der Mittelwert für die blaue Line 10 (10 Impressions * 25 Tage / 25 Tage), für die grüne Linie hingegen 8,06 ((10 Impressions * 15 Tage + 0 Impressions * 6 Tage) / 31 Tage). Dies müssen Sie daher immer berücksichtigen, wenn Sie mit den Rohdaten aus der GSC arbeiten.

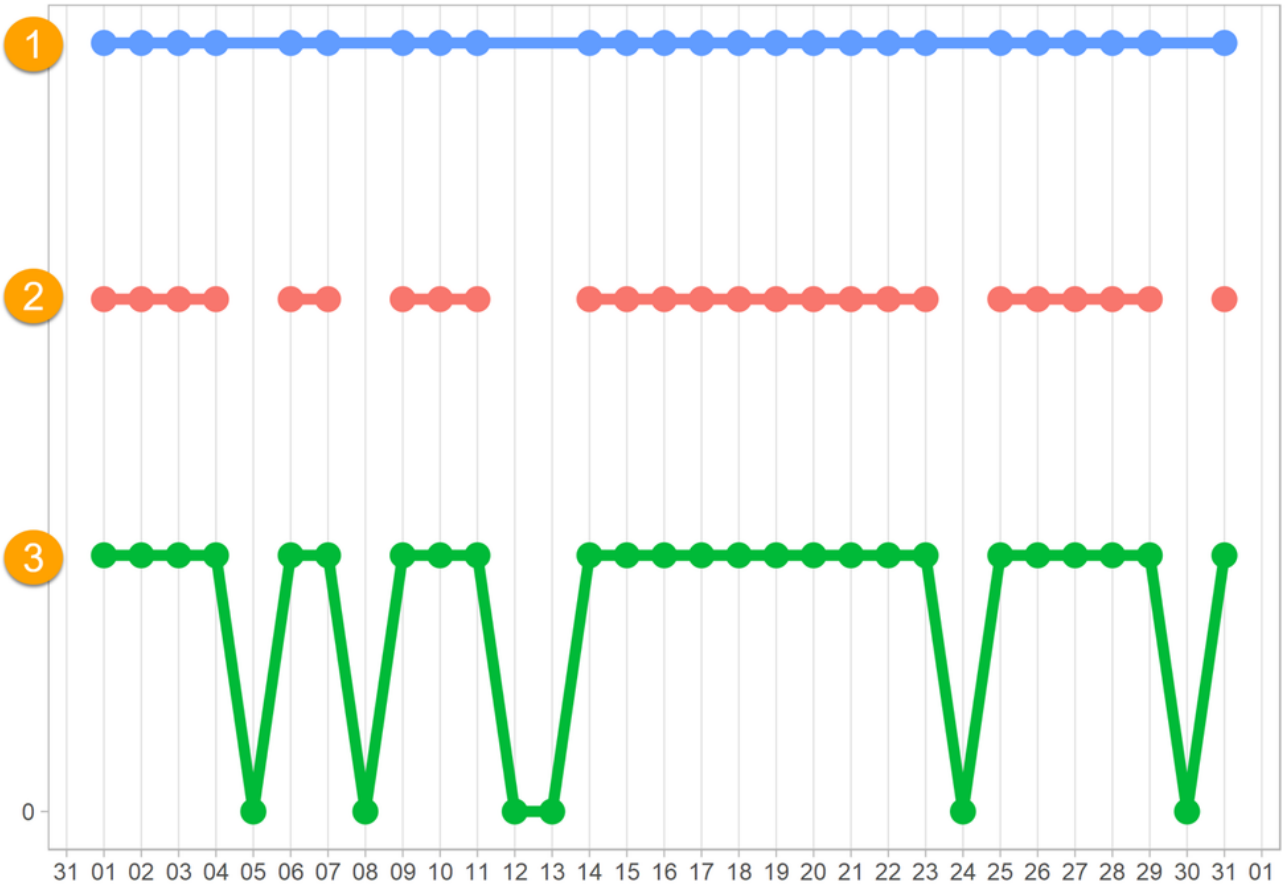


Abbildung 4: Fehlende Datumspunkte und Werte auffüllen, um Verfälschungen des Graphen zu vermeiden

Zurück zum Skript: Nachdem Sie die Packages geladen haben, fügen Sie den Code-Abschnitt, wie in Abbildung 5 zu sehen, ein. Der DataFrame *gsa_dim_date*, der die Performance-Daten auf Tagesbasis enthält, wird an die Funktion *pad()* aus dem Package *padr* gegeben. Die Funktion erkennt automatisch die Datumsspalte und füllt die fehlenden Datumspunkte auf. Über die Argumente *start_val* und *end_val* definieren Sie die untere und obere Grenze des aufzufüllenden Berichtszeitraums. Die hierfür verwendeten Variablen *START_DATE* und *END_DATE* sind erneut jene, die Sie bereits bei den API-Abfragen genutzt haben. Nach diesem Schritt liegen alle Datumspunkte vor, allerdings enthalten die Spalten *clicks* und *impressions* noch keine Werte für diese Punkte. Diese geben Sie durch die anschließende Funktion *replace_na()* aus dem Package *tidyverse* an. Der bearbeitete DataFrame wird wieder in die Variable *gsa_dim_date* zurückgeschrieben.

```
136
137 # GSC ------------------------------------------------------------------
138
139 # Impressions / Clicks
140
141 # Fehlende Datumspunkte einfügen, Impressions und
142 # Clicks mit 0 auffüllen
143 gsa_dim_date ← gsa_dim_date %>%
144   pad(start_val = START_DATE,
145       end_val = END_DATE) %>%
146   replace_na(list(clicks = 0,
147                   impressions = 0))
148
```

Abbildung 5: Fehlende Datumspunkte ergänzen und Metriken mit 0 auffüllen

In Abbildung 6 sehen Sie die einzelnen Schritte noch einmal im Detail anhand eines exemplarischen DataFrames mit Daten für eine Woche. (1) sind die Rohdaten, bei denen drei Tage fehlen. Bei (2) wurden mittels *pad()* die fehlenden Datumspunkte ergänzt, die Werte für die Metriken fehlen aber noch (*NA*; not available). In (3) wurden diese Metriken via *replace_na()* mit dem Wert *0* aufgefüllt.

1

	date	impressions	clicks
1	2019-01-15	10	1
2	2019-01-16	20	2
3	2019-01-17	15	1
4	2019-01-19	30	3

2

	date	impressions	clicks
1	2019-01-14	*NA*	*NA*
2	2019-01-15	10	1
3	2019-01-16	20	2
4	2019-01-17	15	1
5	2019-01-18	*NA*	*NA*
6	2019-01-19	30	3
7	2019-01-20	*NA*	*NA*

3

	date	impressions	clicks
1	2019-01-14	0	0
2	2019-01-15	10	1
3	2019-01-16	20	2
4	2019-01-17	15	1
5	2019-01-18	0	0
6	2019-01-19	30	3
7	2019-01-20	0	0

Abbildung 6: Exemplarischer DataFrame mit aufgefüllten Werten

Gleitenden Mittelwert der Impressions und Clicks berechnen

Für die Performance-Daten liegt Ihnen nun der gesamte Berichtszeitraum im DataFrame *gsa_dim_date* auf Tagesbasis vor. Bei der Visualisierung sorgt dies jedoch dafür, dass der Graph stark „zappeln“ wird, wenn bspw. die Werte für die Impressions starken Schwankungen unterliegen. In Abbildung 7 (1) sehen Sie dies für exemplarische Daten eines Jahres. Hier einen Trend zu erkennen, ist somit sehr schwer, da er im Rauschen des Graphen

untergehen kann. Ein gängiges Mittel, um den Graphen zu glätten, ist daher, den gleitenden Mittelwert für ein Zeitfenster von sieben Tagen zu berechnen (2).

Hinweis

Wie Sie die Daten visualisieren, erfahren Sie in einem späteren Teil. Hier dienen die Plots zunächst nur der Veranschaulichung, warum die Daten wie beschrieben transformiert werden.

Abbildung 7: Plotting der Rohdaten versus gleitenden Mittelwert

Sie müssen sich dazu vorstellen, dass von einer gegebenen Zeitreihe fortschreitend sieben Tage betrachtet werden und für diese der Mittelwert gebildet wird. In Abbildung 8 sehen Sie dies beispielhaft für ein Zeitfenster von drei Tagen. Zunächst wird der Mittelwert für die Tage 1 bis 3 berechnet, dann für die Tage 2 bis 4 usw.

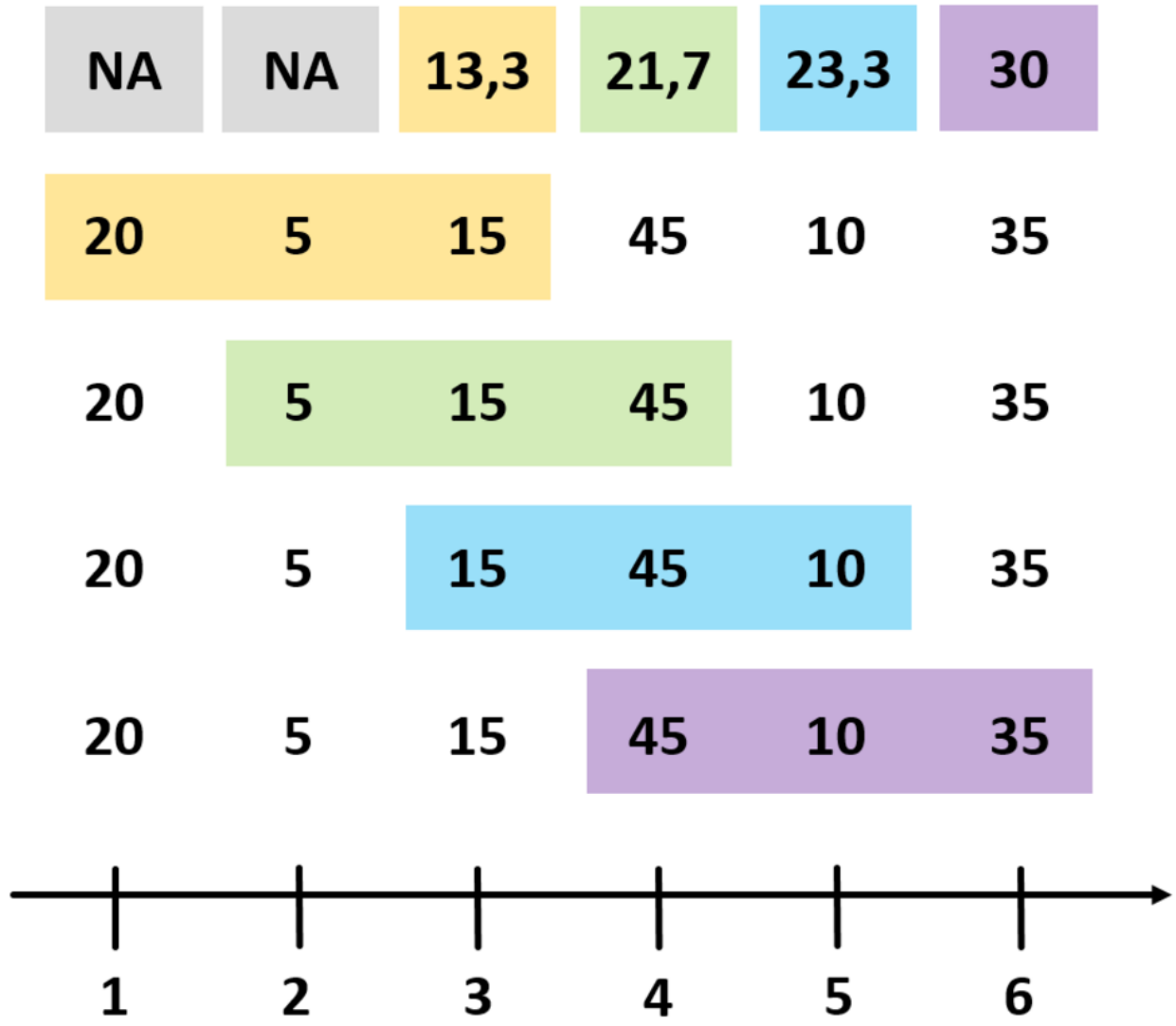


Abbildung 8: Exemplarische Berechnung des gleitenden Mittelwerts für ein Zeitfenster von drei Tagen

Den gleitenden Mittelwert berechnen Sie mit dem Code-Abschnitt, den Sie in Abbildung 9 sehen. Der DataFrame *gsa_dim_date* wird an die Funktion *mutate()* gegeben, die eine bestehende Spalte bearbeiten oder eine neue Spalte hinzufügen kann. In diesem Fall werden die Spalten *rollmean_clicks* und *rollmean_impressions* hinzugefügt. Die Werte der beiden Spalten werden mit der Funktion *rollmean()* aus dem Package *zoo* berechnet. Das erste Argument *x* nimmt die Spalte auf, auf deren Basis der gleitende Mittelwert berechnet werden soll. Das Argument *k* definiert das Zeitfenster – hier sieben Tage. Mit *fill* kann der Wert angegeben werden, der in die Zellen geschrieben werden soll, für die aufgrund der Reduktion der Daten keine Werte mehr vorliegen. Das sehen Sie ebenfalls in

Abbildung 8. Für die Datumspunkte 1 und 2 kann kein Mittelwert berechnet werden, da das Zeitfenster ansonsten links über den Rand des Datumsbereichs hinauslaufen würde. Entsprechend werden diese beiden Datumspunkte mit *NA*s aufgefüllt. Das Argument *align* steuert dieses Verhalten. Es zeigt an, ob der berechnete Mittelwert am rechten Rand des Berichtszeitraums endet und demnach links mit *NA*s aufgefüllt werden soll oder ob am Datumspunkt 1 mit dem Mittelwert begonnen wird und somit die NAs am rechten Rand eingetragen werden.

Für die exemplarischen Daten der Abbildung 8 sähe die finale Tabelle wie in Abbildung 10 dargestellt aus.

```
149
150  # Gleitenden Mittelwert berechnen
151  gsa_dim_date ← gsa_dim_date %>%
152    mutate(rollmean_clicks = rollmean(x = clicks,
153                                      k = 7,
154                                      fill = NA,
155                                      align = "right"),
156           rollmean_impressions = rollmean(x = impressions,
157                                           k = 7,
158                                           fill = NA,
159                                           align = "right"))
160
```

Abbildung 9: Berechnung des gleitenden Mittelwerts für Clicks und Impressions

	date	impressions	rollmean
1	1	20	*NA*
2	2	5	*NA*
3	3	15	13.33333
4	4	45	21.66667
5	5	10	23.33333
6	6	35	30.00000

Abbildung 10: Beispieltabelle mit berechnetem gleitendem Mittelwert

Umwandlung des DataFrames von einem weiten in ein langes Format

Damit folgt der letzte Schritt zum Aufräumen der Performance-Daten. Ihr DataFrame sieht zum jetzigen Zeitpunkt wie in Abbildung 11 aus. Man spricht in diesem Fall von einer weiten Tabelle, denn je Datumspunkt liegen die einzelnen Metriken (*clicks*, *impressions*, *ctr* etc.) als Spalten vor. Für die Visualisierung mit *ggplot* müssen die Daten jedoch in einem langen Format vorliegen. Konkret bedeutet dies, dass die Benennungen der bisherigen Spalten die Werte einer neuen Spalte bilden. Die Tabelle wird quasi gedreht, wobei die Datumsspalte als Ankerpunkt dient. Das umgekehrte Verfahren, die Überführung einer langen in eine weite Tabelle, kennen Sie wahrscheinlich aus Excel, wenn Sie dort eine Pivot-Tabelle erstellen.

Das war jetzt sehr abstrakt: Eine schematische Darstellung mit

nur vier Metrik-Spalten sehen Sie daher in Abbildung 12.

	date	clicks	impressions	ctr	position	rollmean_clicks	rollmean_impressions
1	2018-07-01	2	66	0.03030303	18.136364	*NA*	*NA*
2	2018-07-02	2	41	0.04878049	33.390244	*NA*	*NA*
3	2018-07-03	1	57	0.01754386	35.877193	*NA*	*NA*
4	2018-07-04	0	73	0.00000000	36.054795	*NA*	*NA*
5	2018-07-05	2	98	0.02040816	39.357143	*NA*	*NA*
6	2018-07-06	4	76	0.05263158	44.157895	*NA*	*NA*
7	2018-07-07	6	75	0.08000000	40.266667	2.428571	69.42857
8	2018-07-08	2	53	0.03773585	27.188679	2.428571	67.57143
9	2018-07-09	5	54	0.09259259	29.203704	2.857143	69.42857
10	2018-07-10	2	77	0.02597403	32.116883	3.000000	72.28571
11	2018-07-11	2	72	0.02777778	23.625000	3.285714	72.14286

Abbildung 11: DataFrame gsc_dim_date mit den ergänzten Spalten für die gleitenden Mittelwerte

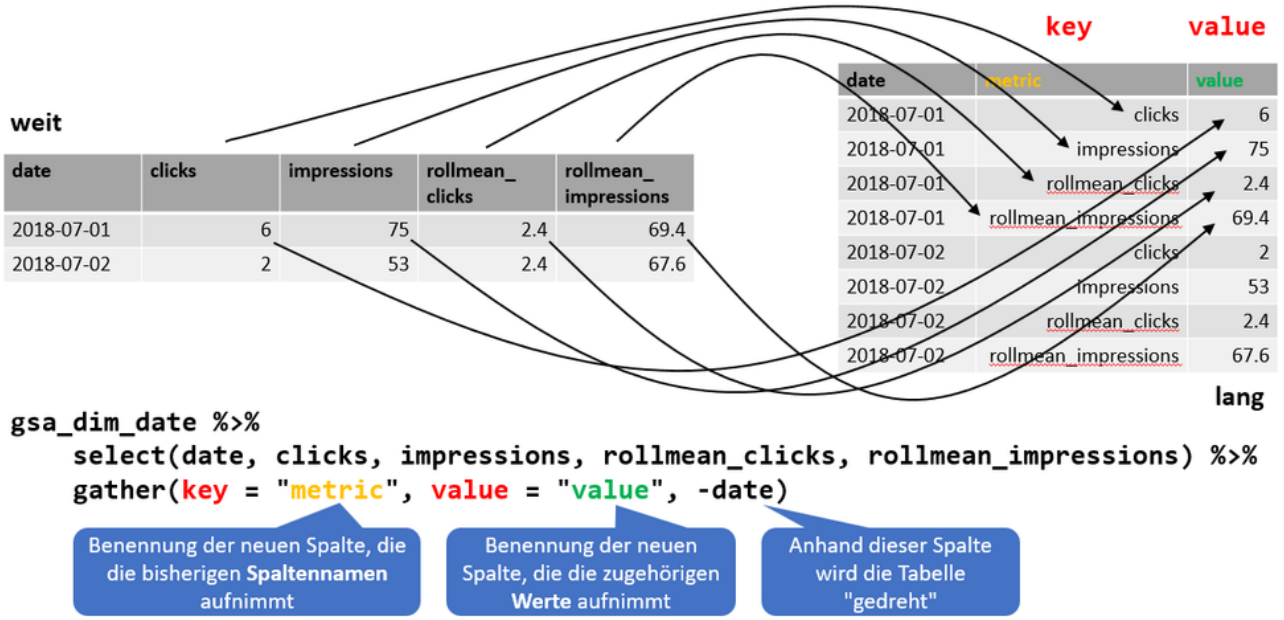


Abbildung 12: Umwandlung einer weiten in eine lange Tabelle mittels gather()

Ihren *gsa_dim_date* DataFrame überführen Sie mit dem Code aus Abbildung 13. *select()* dient dazu, nur die Spalten *date*, *rollmean_clicks* und *rollmean_impressions* beizubehalten. Die restlichen Spalten *clicks*, *impressions*, *ctr* und *position* löschen Sie somit aus dem DataFrame, da sie in der späteren Visualisierung nicht genutzt werden. Die Funktion *gather()* transponiert dann den DataFrame. Mit dem Argument *key* geben Sie den Namen der neuen Spalte an, in die die bisherigen

Spaltenbenennungen als Werte überführt werden. Das Gleiche machen Sie mit dem Argument *value* für die neue Spalte, die die Werte der jeweiligen Metrik-Spalten aufnimmt. Damit haben Sie das Aufräumen der GSC-Daten erledigt. Ihr DataFrame sieht nun wie in Abbildung 14 aus. Die DataFrames mit den Queries (*gsa_dim_date_query*) resp. den URLs (*gsa_dim_date_page*) werden Sie im nächsten Teil dieser Reihe aufbereiten.

```
161
162  # Tabelle transponieren
163  gsa_dim_date ← gsa_dim_date %>%
164    select(date, rollmean_clicks, rollmean_impressions) %>%
165    gather(key = "metric",
166           value = "value",
167           -date)
168
```

Abbildung 13: Überführen des weiten DataFrames gsa_dim_date in ein langes Format

	date	metric	value
1	2018-07-01	rollmean_clicks	*NA*
2	2018-07-02	rollmean_clicks	*NA*
3	2018-07-03	rollmean_clicks	*NA*
4	2018-07-04	rollmean_clicks	*NA*
5	2018-07-05	rollmean_clicks	*NA*
6	2018-07-06	rollmean_clicks	*NA*
7	2018-07-07	rollmean_clicks	2.428571
8	2018-07-08	rollmean_clicks	2.428571
9	2018-07-09	rollmean_clicks	2.857143
10	2018-07-10	rollmean_clicks	3.000000

Abbildung 14: Die ersten Zeilen des langen DataFrames gsa_dim_date

Google-Analytics-Daten vervollständigen

Analog zu den Performance-Daten aus der GSC müssen Sie noch die Google-Analytics-Daten (GA) aufräumen. Auch bei diesen beginnen Sie damit, fehlende Datumswerte zu ergänzen. Insbesondere bei den GA-Daten ist dies eine reine Vorsichtsmaßnahme, da Sie diese für die Channels *Organic Search* und *Direct* abgefragt haben. Dass an einem Tag keine organischen oder direkten Sessions auf Ihrer Website erfolgen, ist höchst unwahrscheinlich – kann aber nichtsdestotrotz bei sehr kleinen bzw. neuen Websites vorkommen. Um sich auf den später generierten Report verlassen zu können, empfiehlt es sich daher, immer für eine „saubere“ Datenbasis zu sorgen.

Fügen Sie den Code aus Abbildung 15 in Ihr Skript ein. Der Unterschied zum Code für die GCS-Daten aus Abbildung 5 besteht hier in dem zusätzlichen Argument *group* der Funktion *pad()*. Sie erinnern sich: Diese ergänzt fehlende Datumspunkte. Die Besonderheit des GA-DataFrames besteht darin, dass dieser bereits im langen Format von der API zurückkommt (Abb. 16). Er besteht nur aus den drei Spalten *date*, *channelGrouping* und *sessions*. Die Spalte *channelGrouping* ist vergleichbar mit Spalte *metric* im DataFrame *gsa_dim_date*. Sie qualifiziert die nebenstehenden Werte in der Spalte *sessions* (vergleichbar mit *value* im GSC-DataFrame) nach den Groups *Direct* und *Organic Search*. Das Argument *group* ist daher nötig, um die fehlenden Datumspunkte individuell für die beiden Ausprägungen *Direct* und *Organic Search* zu ermitteln. In der Abbildung 16 sehen Sie bspw., dass am 2018-07-01 zwar eine *Direct*-Session erfolgt ist, für *Organic Search* jedoch kein Wert vorliegt. Für 2018-07-02 verhält es sich genau andersherum. Für Ersteres muss somit der *Organic-Search*-, für Letzteres der *Direct*-Wert ergänzt werden (Abb. 17).

```r

# Google Analytics ----------------------------------------------------

# Fehlende Datumspunkte einfügen und Sessions mit 0 auffüllen
ga_sessions ← ga_sessions %>%
  pad(start_val = START_DATE,
      end_val = END_DATE,
      group = "channelGrouping") %>%
  replace_na(list(sessions = 0))

```

Abbildung 15: Fehlende Datumspunkte ergänzen und Sessions mit 0 auffüllen

	date	channelGrouping	sessions
1	2018-07-01	Direct	1
2	2018-07-02	Organic Search	2
3	2018-07-03	Organic Search	1
4	2018-07-04	Direct	1
5	2018-07-05	Direct	1
6	2018-07-05	Organic Search	1
7	2018-07-06	Direct	1

Abbildung 16: Langer DataFrame ga_sessions mit teilweise fehlenden Kombinationen aus date und channelGrouping

	date	channelGrouping	sessions
1	2018-07-01	Direct	1
2	2018-07-01	Organic Search	0
3	2018-07-02	Direct	0
4	2018-07-02	Organic Search	2
5	2018-07-03	Direct	0
6	2018-07-03	Organic Search	1
7	2018-07-04	Direct	1
8	2018-07-04	Organic Search	0
9	2018-07-05	Direct	1
10	2018-07-05	Organic Search	1

Abbildung 17: DataFrame ga_sessions mit ergänzten Kombinationen aus date und channelGrouping

Gleitenden Mittelwert der GA-Sessions berechnen

Im Anschluss errechnen Sie den gleitenden Mittelwert für die Sessions (Abb. 18). Auch hier müssen Sie dies separat für die beiden Channels machen. Daher erfolgt in der zweiten Zeile ein *group_by()* der Spalte *channelGrouping*, sodass die nachfolgenden Funktion *rollmean()* separat auf diese beiden Gruppen angewendet wird.

```
179
180  # Gleitenden Mittelwert berechnen
181  ga_sessions ← ga_sessions %>%
182    group_by(channelGrouping) %>%
183    mutate(rollmean_sessions = rollmean(x = sessions,
184                                        k = 7,
185                                        fill = NA,
186                                        align = "right"))
187
```

Abbildung 18: Berechnung des gleitenden Mittelwerts der Sessions je channelGrouping

Fazit

Gratulation! Wenn Sie es bis hierher geschafft haben, sind Sie schon ein gutes Stück in die Gefilde der Datentransformation vorgedrungen. Sie haben das Wehklagen eines jeden Daten-Analysten nachvollziehen gelernt, dass Analysen zu 80 % aus Datenbereinigung und nur zu 20 % aus der eigentlich spannenden Analysetätigkeit bestehen.

Darüber hinaus haben Sie erfahren, wie Sie fehlende Datumswerte in Zeitreihen vervollständigen, um Verfälschungen bei Datenaggregationen zu vermeiden. Außerdem haben Sie den gleitenden Mittelwert verschiedener Metriken berechnet, um Graphen zu glätten und somit Trends besser erkennbar zu machen. Last but not least haben Sie ein mächtiges Werkzeug an die Hand bekommen, um Tabellen zu transponieren und sie so in ein handhabbares langes Format zu bringen.

Lesen Sie im dritten Teil in der nächsten Ausgabe, wie Sie Ihr SEO-Reporting mit R automatisieren können. So lassen sich z. B. mit wenig Aufwand die Top-SEO-Seiten nach Klicks und die Top-Suchanfragen aus der Google Search Console ermitteln und für Berichte in speziellen DataFrames zusammenfassen.

Title-Geddon – war da was bei Ihnen?

R-Leuchtungen! Teil 9

R-Leuchtungen! Teil 9: Title-Geddon – war

da was bei Ihnen? – websiteboosting.com

In den Ausgaben bis 54 bis 58 der Website Boosting konnten Sie in der Serie „R4SEO“ von Patrick Lürwer nachvollziehen, wie man die kostenlose Software R verwendet, was sie leistet und wie man sie nutzbringend für die eigene Arbeit für SEO bzw. die Aufklärung im Online-Marketing einsetzen kann. R…

In den Ausgaben bis 54 bis 58 der Website Boosting konnten Sie in der Serie „R4SEO“ von Patrick Lürwer nachvollziehen, wie man die kostenlose Software R verwendet, was sie leistet und wie man sie nutzbringend für die eigene Arbeit für SEO bzw. die Aufklärung im Online-Marketing einsetzen kann. R wurde ja ursprünglich für Statistik entwickelt. Wer das bisher als Entschuldigung verwendet hat, es deswegen links liegen zu lassen, dem sei versichert, dass er damit komplett falschliegt. R kann für den Einsatz im Unternehmen, allen Bereichen voran „Online“, sehr viel mehr leisten, als statistische Berechnungen durchzuführen. Genau genommen ist es ein wirklich nützliches Helferlein bei allen Aufgaben im Umgang mit größeren Datenmengen, mit Daten, die man erst in eine gewisse Struktur bringen muss, und bei der automatischen oder halb automatischen Datenbeschaffung aus praktisch fast allen Quellen aus dem Web! Für die interessierten Einsteiger, aber auch für alle, die nach der Serie von Patrick Lürwer „R-Blut“ geleckt haben, startete in der Ausgabe 62 die neue anwendungsorientierte Serie „R-Leuchtungen“. Sie werden in jeder Ausgabe erfahren, wie Sie ohne Programmierkenntnisse jeweils ein definiertes und in der Online-Praxis häufiger auftretendes Problem rund um das Thema Daten und Auswertungen lösen können. Und keine Sorge, die kleinen Hilfe-Tutorials nehmen Sie Schritt für Schritt an der Hand, sodass Sie auch als Neuling von der Power von R profitieren können. Was hält Sie also ab, das einfach mal auszuprobieren? Die einzelnen Schritte müssen Sie übrigens nicht im Detail verstanden haben. Um an die hilfreichen Daten für ein besseres Ranking zu kommen, müssen Sie im Prinzip nur nachmachen, was Sie hier beschrieben finden. Für dieses Beispiel brauchen Sie übrigens

keinerlei kostenpflichtige Tools.

Worum geht es diesmal?

Um den 21.08.2021 herum hat Google das sogenannte Title-Update ausgerollt. Dabei wurde in den Suchergebnissen bei nicht wenigen Seiten der im jeweiligen Quellcode hinterlegte Title durch einen von Google generierten Title ersetzt. Das führte zu einigem Aufruhr bei den aufmerksamen Sitebetreibern, weil zum Teil wohl weniger passende Title erzeugt wurden und man zum anderen einen gewissen Kontrollverlust verspürte.

Einige Tools, so z. B. die SISTRIX-Toolbox, zeigen bereits für einzelne Domains an, ob und welche Title getauscht wurden. Ebenso gibt es bereits Browser-Plug-ins, die Veränderungen direkt in den Suchergebnissen sichtbar machen. Das alles sind aber nur punktuelle Hinweise und bisher bekommt man kaum weiter verarbeitbare Daten geliefert bzw. kann die Ergebnisse nicht downloaden für Checklisten und Handlungsanweisungen.

Die Kernfrage ist, ob sich bei (nahezu) gleichbleibender Position eines Keywords die CTR, also die Klickrate, signifikant derart verändert hat, dass sich der Grund beim Title-Tausch durch Google vermuten lässt. Bleibt z. B. vor und nach dem Update die Position 3 stabil stehen, aber die Klickrate sinkt von 15 % auf 5 %, dann könnte mit hoher Wahrscheinlichkeit das veränderte Ergebnissnippet dafür verantwortlich sein, da sich sonst ja nichts verändert hat. Würde die Position steigen oder fallen, würde sich in jedem Fall bekanntlich auch immer die CTR ändern. Insofern gilt es, Rankings zu filtern, deren Position sich nicht stark verändert hat, jedenfalls nicht über eine normale Schwankungsbreite hinaus. Natürlich tritt auch bei der CTR eine gewisse Schwankung auf.

Die Aufgabe lautet also: Finde alle Rankings, deren Position gleich geblieben ist UND deren CTR sich deutlich verändert hat. Dabei sollten alle Suchphrasen, die als Besonderheit den

Domainnamen (Brandbegriffe) enthalten, ignoriert werden.

Das Skript in der vorliegenden Ausgabe nimmt sich dieses Problems an und ist wie folgt strukturiert:

1. Abholen aller Rankingdaten für zwei definierbare Zeitfenster (vor und nach dem Update)
2. Definieren gewisser Schwankungstoleranzen für die CTR
3. Definieren gewisser Schwankungstoleranzen für die Position
4. Angabe der Brand-Suchbegriffe (Domainname)
5. Filtern aller ausgeschlossenen Fälle von Punkt 2 bis 4.
6. Ausgabe der übrig gebliebenen Daten in ein CSV-File zur weiteren Verwendung

Um tatsächlich umfassend mit den Rohdaten direkt von Google arbeiten zu können, muss ein sog. (kostenloses) Entwicklertoken von Google vorliegen (siehe den Infokasten am Rand).

Ihr API-Account bei Google

Um automatisiert bzw. computergesteuert Daten von Google-Tools holen zu können, brauchen Sie dort einen „Developer-Account“. Für diesen bekommen Sie eine von Google erzeugte Mailadresse und ein File mit der Endung .json für die Authentifizierung Ihres Computers bzw. für R, um Daten zu holen. Die Mailadresse dient dazu, einen Nutzer für die Search Console anlegen zu können, und mit dem File kann sich eine Maschine gegenüber Google als berechtigt ausweisen. Wie Sie einen solchen Account anlegen können, haben wir in der Ausgabe 65 ausführlich beschrieben. Falls Sie den Account da angelegt haben, können Sie jetzt erneut darauf zugreifen und müssen nichts weiter tun.

Haben Sie das übersprungen, müssen und sollten Sie das (Schritt 1 und 2 im oben genannten Beitrag) unbedingt

nachholen. Bereits in den vergangenen Ausgaben hatten wir Ihnen immer wieder Skripts an die Hand gegeben, mit denen Sie wirklich sehr nützliche Auswertungen mit echten Google-Daten ganz einfach mit wenigen Klicks erzeugen können. Ohne diesen Account bekommen Sie leider keine ausführlichen Daten. Und ohne diese Daten können Sie natürlich auch keine eigenen und detaillierten Hinweise auf Optimierungsmöglichkeiten für Ihr Ranking erzeugen.

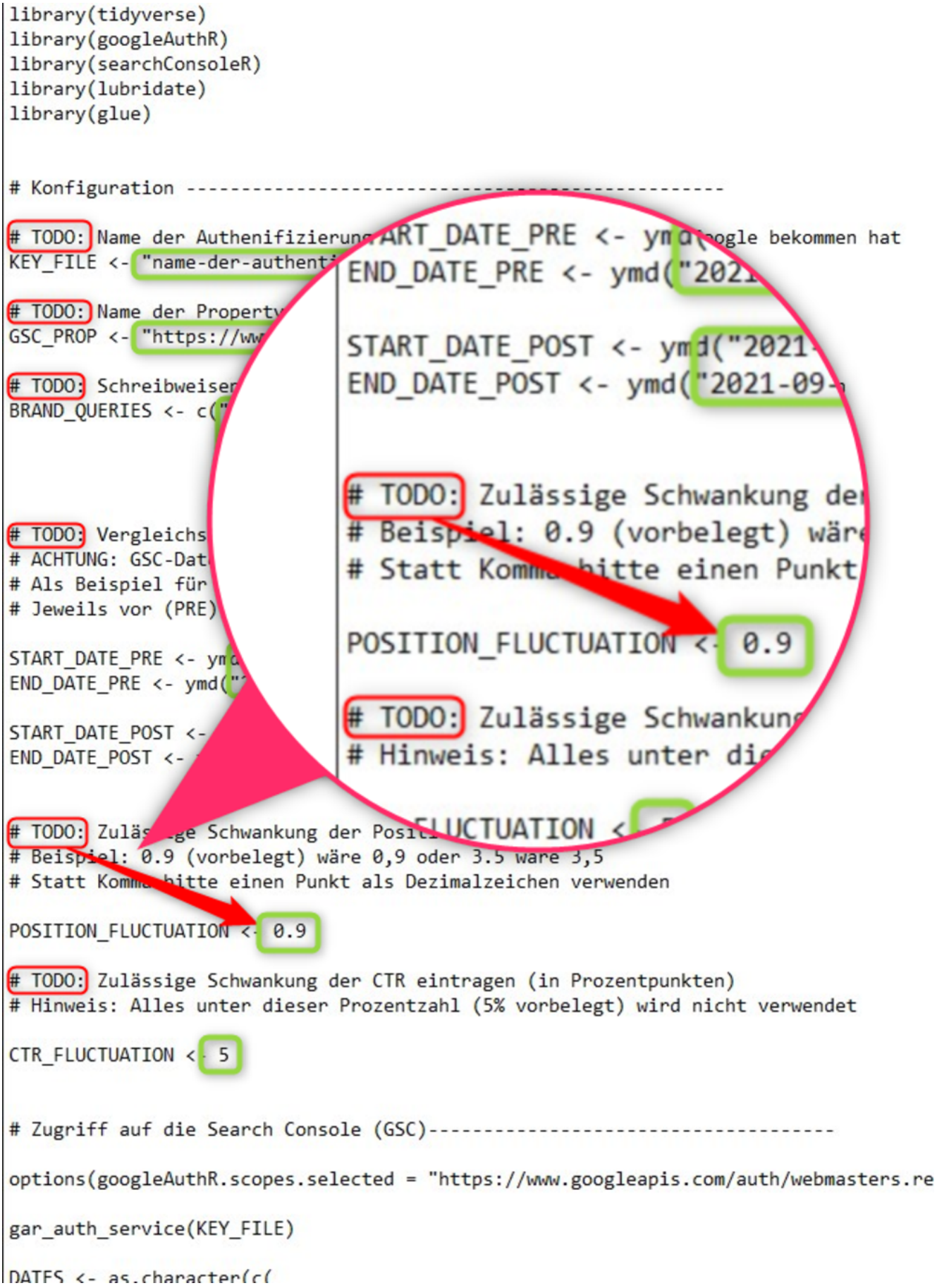


Abbildung 1: An einigen Stellen im Code („# TODO“) müssen einfache Anpassungen vorgenommen werden

Schritt 1: Das R-Skript downloaden und die Parameter eingeben

Holen Sie sich unter einfach.st/rcode6 das Skript für diese Ausgabe 70 bzw. kopieren Sie von dort ganz einfach den Code in Ihre Zwischenablage. Starten Sie R-Studio als „Benutzerhülle“ für R. Mit File/New File erhalten Sie links oben ein neues, noch leeres Fenster für den R-Code. Kopieren Sie ihn von der Webseite direkt in dieses Fenster.

Stellen Sie sicher, dass R mit dem richtigen Arbeitsverzeichnis (Working Directory, kurz WD) arbeitet bzw. legen Sie das Verzeichnis auf Ihrem Computer fest, das für diese Session verwendet werden soll. Dazu klicken Sie sich einfach rechts unten in dem Fenster wie in Abbildung 2 unter den beiden Ziffern 1 gezeigt zu dem gewünschten Verzeichnis durch. Achtung: In diesem Verzeichnis muss dann auch die Datei (oder eine Kopie davon) von Google liegen, die man zu Authentifizierung des Computers braucht. Sie endet auf .json und enthält vorne die bei Google angegebene Mailadresse, gefolgt von einer Zahlen-/Buchstabenreihenfolge. Sie sieht in etwa so aus:

mein-accountname-cr4436ad9828.json

Stimmt das angezeigte Verzeichnis, klickt man einfach unter dem Zahnradsymbol „More“ auf „Set As Working Directory“. Im Fenster links daneben wird dies nun bestätigt mit:

```
> setwd("~/was/sie/ausgewaehlt/haben")
```

Wie immer müssen für R dann noch bestimmte Funktionsbibliotheken (Librarys) installiert bzw. aufgerufen werden. Nach dem erstmaligen Installieren mit dem Befehl

```
install.packages("libraryname")
```

behält R diese Bibliothek in ihrem Dateispeicherbereich. Sie muss dann nicht mehr installiert werden. Der Aufruf einer

Bibliothek, um damit arbeiten zu können, bzw. das Laden in den aktiven Arbeitsspeicher muss hingegen nach jedem Start von R vollzogen werden. Der Befehl dazu lautet:

```
library(libraryname)
```

Es gilt zu beachten, dass hier keine Anführungszeichen wie beim Installieren verwendet werden. Um den Aufruf einer Bibliothek in den Arbeitsspeicher muss man sich nicht aktiv kümmern, weil er in der Regel immer am Anfang im Skript steht und daher sowieso immer mit ausgeführt wird. Wer die R-Leuchtungen der letzten Ausgaben aktiv angewendet hat, hat bereits alle nötigen Bibliotheken installiert und kann den Code daher später einfach durchlaufen lassen. Bibliotheken kann man übrigens auch leicht über den Menüpunkt „Tools“ und dort „Install Packages“ einzeln installieren. R sucht sich die Speicherorte im Web von alleine.

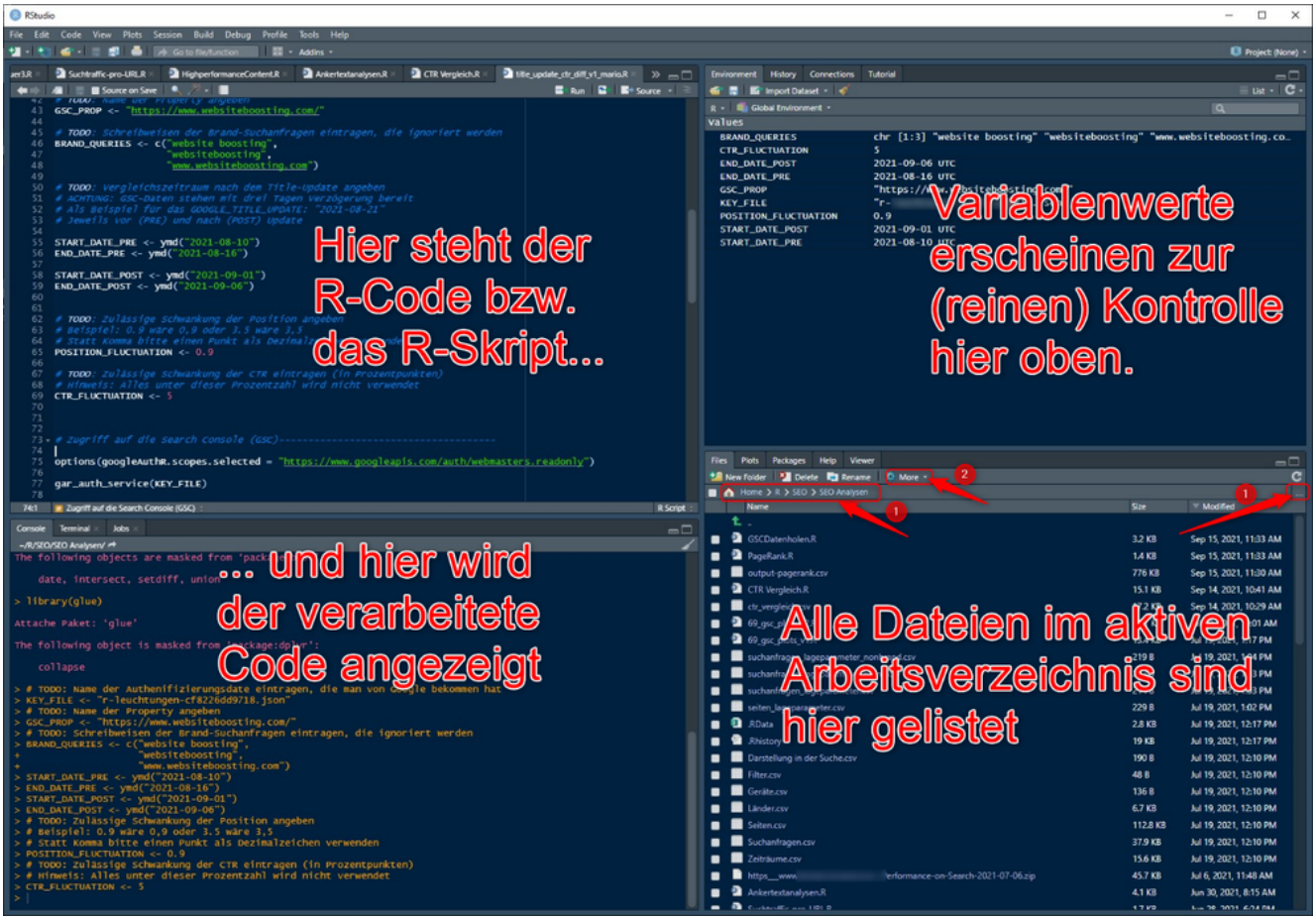


Abbildung 2: So ist R-Studio strukturell aufgeteilt

Die Anpassungen im Skript sind dort nochmals dokumentiert (alle Zeilen mit einem führenden „#“ stellen Kommentarzeilen

dar und sind kein Code).

Sie müssen an sechs Stellen Eingaben/Änderungen vornehmen. Diese sind alle mit „#TODO“ gekennzeichnet.

Zunächst müssen Sie den genauen Namen Ihrer Authentifizierungsdatei (die mit .json am Ende) angeben. Überschreiben Sie einfach die Vorgabe entsprechend:

```
KEY_FILE <- "name-der-authentifizierungsdatei.json"
```

Anschließend müssen Sie Ihren Domainnamen noch hinterlegen bzw. überschreiben:

```
GSC_PROP <- https://www.domainname.com/
```

Optional: Möchten Sie Suchanfragen nach Ihrem Domainnamen/Brand ignorieren? Dann überschreiben Sie einfach ebenfalls die Vorbelegungen. Brauchen Sie mehr Zeilen, kopieren Sie einfach z. B. die Zeile 2 und fügen Sie diese mit einem überschriebenen Suchwort wieder ein. Diese müssen nur jeweils durch Anführungszeichen und ein abschließendes Komma getrennt werden:

```
BRAND_QUERIES <- c("brandbegriff-1",
 "brandbegriff-2",
 "brandbegriff-3",
 "brandbegriff-n")
```

Optional: Nun müssen bzw. können Sie noch die beiden Zeiträume für den Vergleich definieren. Vorbelegt ist:

```
START_DATE_PRE <- ymd("2021-08-10")
END_DATE_PRE <- ymd("2021-08-16")

START_DATE_POST <- ymd("2021-09-01")
END_DATE_POST <- ymd("2021-09-06")
```

PRE steht für „vor dem Update“ und POST entsprechend für einen Zeitraum danach. Das Startdatum ist im Beispiel hier so gewählt, dass sicherheitshalber ein Zeitraum einige Tage vor dem Title-Update fixiert wurde. Bei Bedarf können Sie beide

Zeiträume entsprechend anpassen, sollten z. B. neue Updates das nötig erscheinen lassen.

Nicht nur für Title-Updates!

Mit dem vorliegenden Skript können Sie übrigens jede Art von Klickveränderungen Ihrer Suchphrasen „überwachen“ bzw. prüfen. Da Sie den Start- und Endzeitraum des Vergleichs frei wählen können, lässt sich damit prinzipiell auch jeder Zeitraum mit jedem anderen vergleichen! Damit haben Sie für jede Art von Sonderanalyse ein recht machtvolles kleines Skript an der Hand, das Ihnen in wenigen Sekunden datengestützte Antworten liefern kann.

Optional: Legen Sie einen Wert fest, um den die Position schwanken darf, ohne gleich als echte Positionsveränderung zu gelten. Da Google Kommawerte bzw. die durchschnittliche Position angibt, schwankt diese eigentlich fast immer um einige Prozentpunkte hinter dem Komma (das in R mit einem Punkt gesetzt wird). Würde man diese Schwankung nicht berücksichtigen, gäbe es für jedes Keyword eine Positionsveränderung, weil bereits ein Ansteigen von Position 2,3 auf 2,4 als Änderung gelten würde. Vorbelegt ist der Wert 0,9, aber Sie können diesen Ihren Bedürfnissen entsprechend verändern. Das macht dann Sinn, wenn Sie zu viele oder zu wenige Daten in der Auswertung haben. Da dies von Domain zu Domain schwankt, kann man keine generellen, verbindlichen Werte angeben. Hier müssen Sie ggf. etwas experimentieren.

```
POSITION_FLUCTUATION <- 0.9
```

Optional: Ebenso macht es Sinn, einen natürlichen Korridor für Schwankungen im CTR-Wert zu berücksichtigen. Hier wurde 5 % vorbelegt. Das bedeutet, dass eine Schwankung unter oder über 5 % noch nicht als nennenswerte Veränderung gesehen wird und die Daten weggefiltert werden, die sich innerhalb dieser Breite befinden. Sollten Sie zu viele oder zu wenig Daten bekommen, versuchen Sie einfach, das Skript mit veränderten

Werten noch mal laufen zu lassen.

```
CTR_FLUCTUATION <- 5
```

Das war auch schon alles, was Sie hinterlegen bzw. anpassen müssen. Klicken Sie nun einfach auf „Source“ rechts oben im linken oberen Fenster (Abbildung 3). Dies löst das Abarbeiten des gesamten Skripts auf einen Rutsch aus.

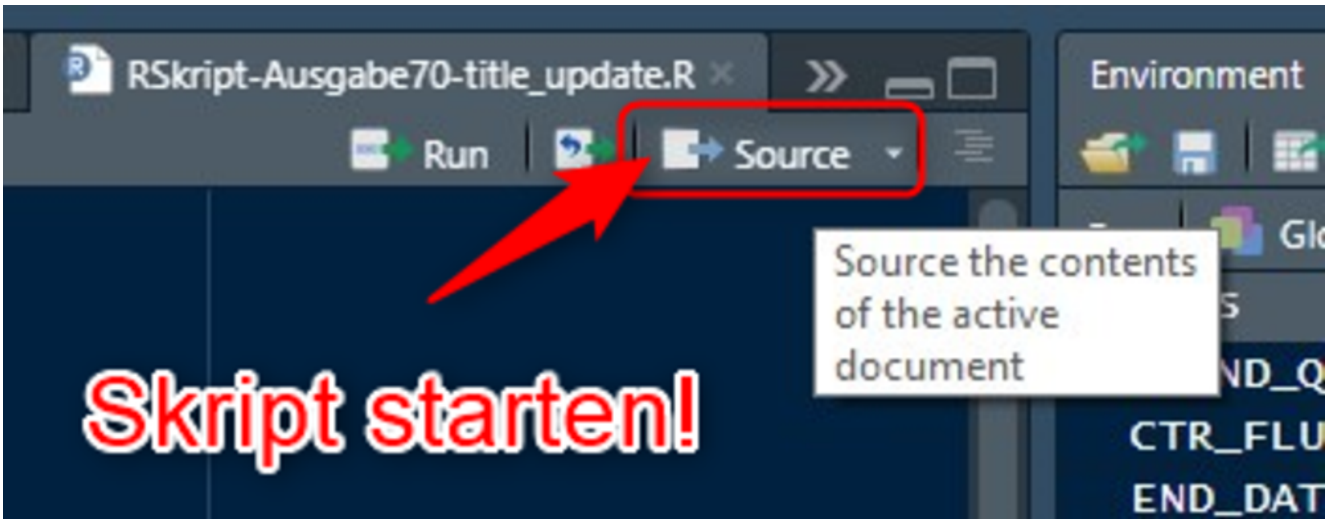


Abbildung 3: Ein Mausklick auf „Source“ startet das komplette Skript

Das Durchlaufen kann je nach Datenumfang etwas dauern, da die Daten von der Search Console im Batchbetrieb für jeden angeforderten Tag (über das Zeitintervall oben definiert) abgeholt werden (Abbildung 4). Anschließend werden alle empfangenen Daten automatisch ohne weiteres Zutun vom Skript verarbeitet.

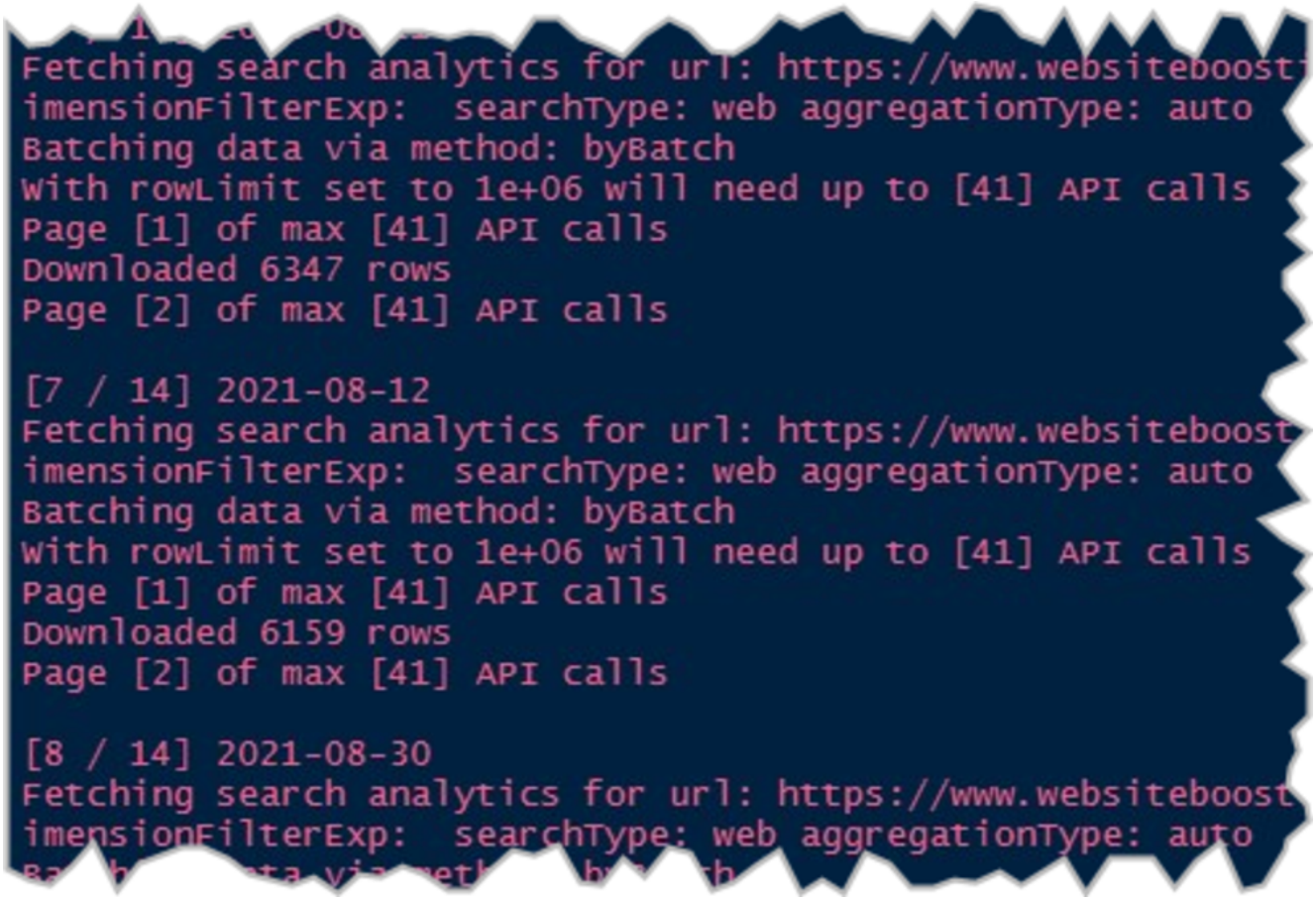


Abbildung 4: So bzw. so ähnlich sieht das laufende Skript im Consolenfenster aus

Am Ende des Skripts sorgt noch die letzte Codezeile dafür, dass die fertigen Daten in einer CSV namens „ctr_vergleich.csv“ im Arbeitsverzeichnis gespeichert werden:

```
write_csv2(gsc_data, "ctr_vergleich.csv")
```

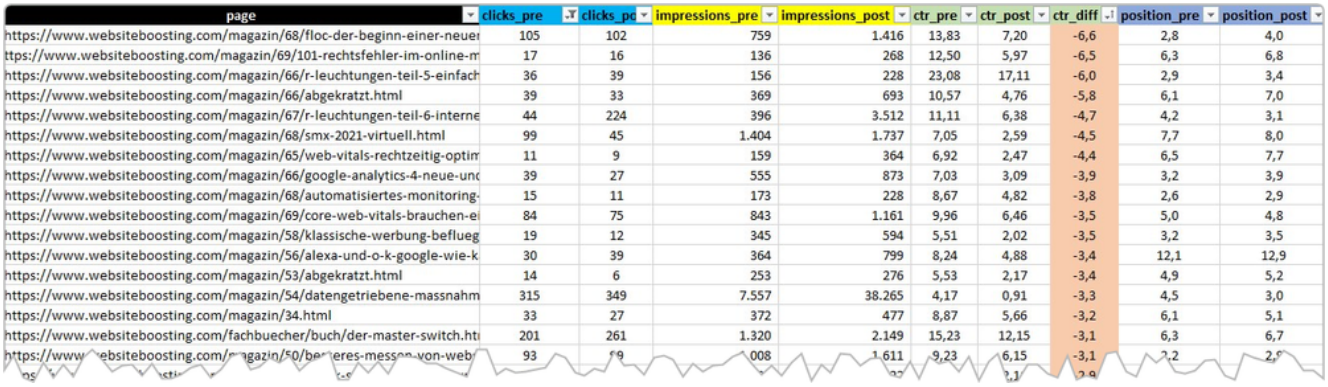
Selbstverständlich können Sie hier auch einen eigenen Dateinamen vergeben, indem Sie die Vorgabe „ctr_vergleich.csv“ im Skript einfach nach Ihren Wünschen verändern.

Schritt 2: Sehen Sie sich die Auswertungen an und arbeiten Sie damit

Wenn Sie die erzeugte Datei öffnen, finden Sie alle URLs, die in dem von Ihnen definierten Zeitraum bei gleichbleibender Position (innerhalb der definierten Schwankungsbreite) eine

CTR-Veränderung über der Prozentzahl haben, die Sie definiert haben. Das ist der Startpunkt für weitere Analysen. Durch die Verwendung der SEO-Tools für Excel kann man die Tabelle problemlos mit weiteren Metriken anreichern, wie z. B. den aktuellen Title und den Inhalt der H1 dazu holen. Dem Vernehmen nach wird beim Austausch der Title von Google häufig auf den Inhalt der H1 zurückgegriffen, sofern er nicht identisch mit dem Title oder wenig aussagekräftig ist.

Beachten Sie bitte, dass Sie von der Google Search Console deutlich mehr Daten bekommen als mit vielen anderen Tools. Es ist daher nicht unwahrscheinlich, dass Sie auch für nur wenige Impressions oder Klicks Datensätze im Ergebnis bekommen. Hier sind die Änderungen der CTR natürlich statistisch nicht aussagekräftig bzw. belastbar. Sie sollten Sie daher in Excel am besten wegfiltern (Zahlenfilter/größer als).



page	clicks_pre	clicks_po	impressions_pre	impressions_post	ctr_pre	ctr_post	ctr_diff	position_pre	position_post
https://www.websiteboosting.com/magazin/68/floc-der-beginn-einer-neuer	105	102	759	1.416	13,83	7,20	-6,6	2,8	4,0
ttps://www.websiteboosting.com/magazin/69/101-rechtsfehler-im-online-m	17	16	136	268	12,50	5,97	-6,5	6,3	6,8
https://www.websiteboosting.com/magazin/66/r-leuchtungen-teil-5-einfach	36	39	156	228	23,08	17,11	-6,0	2,9	3,4
https://www.websiteboosting.com/magazin/66/abgekratzt.html	39	33	369	693	10,57	4,76	-5,8	6,1	7,0
https://www.websiteboosting.com/magazin/67/r-leuchtungen-teil-6-interne	44	224	396	3.512	11,11	6,38	-4,7	4,2	3,1
https://www.websiteboosting.com/magazin/68/smx-2021-virtuell.html	99	45	1.404	1.737	7,05	2,59	-4,5	7,7	8,0
https://www.websiteboosting.com/magazin/65/web-vitals-rechtzeitig-optim	11	9	159	364	6,92	2,47	-4,4	6,5	7,7
https://www.websiteboosting.com/magazin/66/google-analytics-4-neue-und	39	27	555	873	7,03	3,09	-3,9	3,2	3,9
https://www.websiteboosting.com/magazin/68/automatisiertes-monitoring-	15	11	173	228	8,67	4,82	-3,8	2,6	2,9
https://www.websiteboosting.com/magazin/69/core-web-vitals-brauchen-ei	84	75	843	1.161	9,96	6,46	-3,5	5,0	4,8
https://www.websiteboosting.com/magazin/58/klassische-werbung-beflueg	19	12	345	594	5,51	2,02	-3,5	3,2	3,5
https://www.websiteboosting.com/magazin/56/alexa-und-o-k-google-wie-k	30	39	364	799	8,24	4,88	-3,4	12,1	12,9
https://www.websiteboosting.com/magazin/53/abgekratzt.html	14	6	253	276	5,53	2,17	-3,4	4,9	5,2
https://www.websiteboosting.com/magazin/54/datengetriebene-massnahm	315	349	7.557	38.265	4,17	0,91	-3,3	4,5	3,0
https://www.websiteboosting.com/magazin/34.html	33	27	372	477	8,87	5,66	-3,2	6,1	5,1
https://www.websiteboosting.com/fachbuecher/buch/der-master-switch.ht	201	261	1.320	2.149	15,23	12,15	-3,1	6,3	6,7
https://www.websiteboosting.com/magazin/50/bessseres-messen-von-web	93	69	1.008	1.611	9,23	6,15	-3,1	2,2	2,9

Abbildung 5: Formatiert, nach CTR-Differenz sortiert und nach Klicks gefiltert

Titles - Zeige 1 bis 100 von 351

Keyword	16.08.202…	20.09.202…	Titel
kmu webseiten	1	1	Websites in the small - KMU-Webseiten erfolgreich machen - websiteboosting.com Websites in the small - KMU-Webseiten erfolgreich machen
piwik webanalyse anleitung	1	1	Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - websiteboosting ... Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - Website Boosting
pagerank kostenlos	2	2	SEO-Stärke einer Website beurteilen - mit kostenlosen Tools ... SEO-Stärke einer Website beurteilen - mit kostenlosen Tools!
piwik anleitung deutsch	2	2	Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - websiteboosting ... Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - Website Boosting
piwik dokumentation deutsch	4	2	Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - websiteboosting ... Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - Website Boosting
suchmaschinen optimierung typo3	2	2	Suchmaschinenoptimierung für TYPO3 – die Basics ... Suchmaschinenoptimierung für TYPO3 – die Basics
suchmaschinenoptimierung typo3	2	2	Suchmaschinenoptimierung für TYPO3 – die Basics ... Suchmaschinenoptimierung für TYPO3 – die Basics
typo3 suchmaschinenoptimierung	2	2	Suchmaschinenoptimierung für TYPO3 – die Basics ... Suchmaschinenoptimierung für TYPO3 – die Basics
interne verlinkung onpage	3	3	SEO fängt mit OnPage an Teil 3: Interne Verlinkungsstruktur und ... SEO fängt mit OnPage an Teil 3: Interne Verlinkungsstruktur und
suchsysteme	3	3	Alternative Suchsysteme – Trafficquellen abseits von Google ... Alternative Suchsysteme – Trafficquellen abseits von Google?
vorteile piwik	3	3	Web Analytics mit Piwik – die Vorteile eines Tag-Managers (Teil 3 ... Web Analytics mit Piwik – die Vorteile eines Tag-Managers (Teil 3)
piwik deutsch installieren	5	4	Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - websiteboosting ... Web Analytics mit Piwik – so starten Sie durch! (Teil 1) - Website Boosting

Abbildung 6: Einige Tools wie z. B. SISTRIX zeigen veränderte Titles in den Suchsnippets für zwei Datumsangaben an

Bei den meisten Analysen, die wir testhalber vorgenommen haben, hat sich das „Title-Geddon“, wie das Update im Netz zum Teil genannt wurde, übrigens nicht besonders stark ausgewirkt. Es gab zwar möglicherweise viele Title, die tatsächlich getauscht wurden, in eine stark messbare CTR-Veränderung mündete dies allerdings nicht. Die Fälle, die wir extrahieren konnten, zeigten bei einer manuellen Einzelfallprüfung, dass man sicherlich mit der getauschten Zeile gut leben kann.

Trotzdem gibt es im Web zum Teil heftige Beschwerden von Webmaster, dass die Algorithmen alles deutlich verschlimmert hätten. Es ist noch zu früh, stabile Aussagen dazu zu machen, vor allem weil Google angekündigt hat, nochmals nachbessern zu wollen. Trotzdem kann es natürlich nichts schaden, sich einmal in Ruhe anzusehen, ob die Performance Ihrer Suchergebnisse möglicherweise gelitten hat. Hier hilft Ihnen das vorliegende Skript!