Mario Fischer, Patrick Lürwer

R-LEUCHTUNGEN!

Teil 5: Einfache Rankingpotenziale heben

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.

DER AUTOR

Mario Fischer ist Herausgeber und Chefredakteur der Website Boosting und seit der ersten Stunde des Webs von Optimierungsmöglichkeiten fasziniert. Er berät namhafte Unternehmen aller Größen und Branchen und lehrt im neu gegründeten Studiengang E-Commerce an der Hochschule für angewandte Wissenschaften in Würzburg.

DER AUTOR

Patrick Lürwer ist Senior Analyst bei get:traction GmbH. Dort ist er für die Datenerfassung, -aufbereitung und -analyse zuständig. Sein tägliches Handwerkszeug sind R, Python und KNIME.

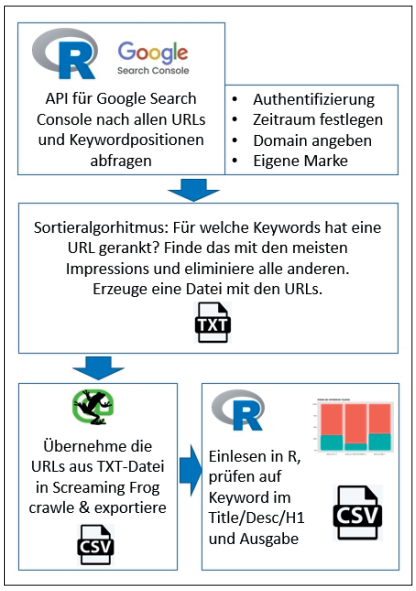


Abb. 1: Übersicht über den Arbeitsablauf bzw. den Aufbau des Skripts

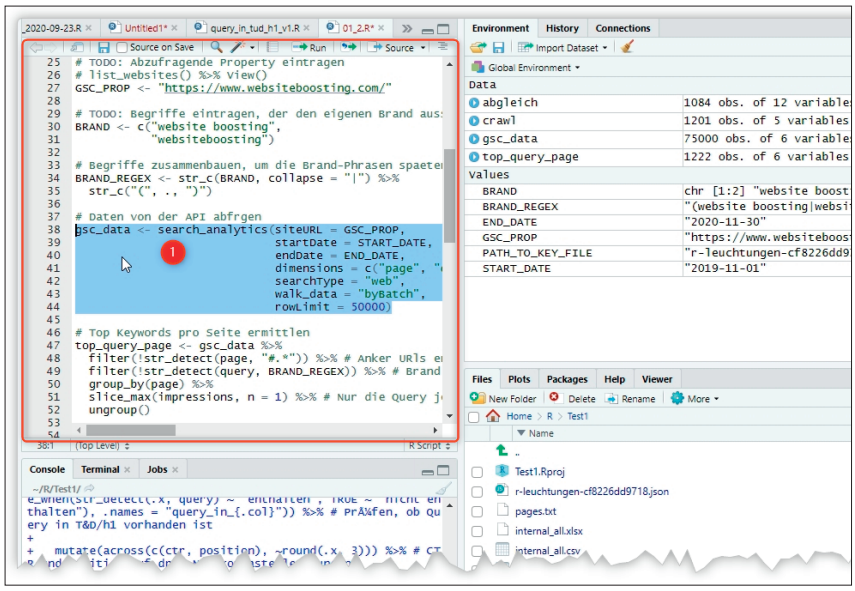


Abb. 2: Tipp: Zeilen oder Abschnitte markieren und STRG+Enter drücken löst -nur- diese Befehlzeilen aus *(Siehe Tippkasten)*

Worum geht es diesmal?

Bekanntlich sind die Inhalte im Titel einer Webseite nach wie vor recht wichtig für das Ranking. Jetzt wäre es doch spannend zu wissen, welche Seiten für Suchbegriffe bereits ranken, aber noch gar nicht im Title vorhanden sind. Ergänzt man diese dann dort, verbessert sich in der Regel das Ranking noch um einige Positionen. Ist das Keyword dann noch in der Description enthalten, wird es fett formatiert mit angezeigt in den Suchergebnissen. Das erhöht die Klickrate und bringt damit direkt mehr Traffic. Viele Experten glauben, dass eine erhöhte Klickrate mittelfristig auch positive Wirkungen auf das Ranking hat (auch wenn die Einträge in der Meta-Description keine direkten Auswirkungen haben). Weiterhin wäre es auch gut (ob rankingrelevant oder nicht), wenn in der ersten Hauptüberschrift, der H1 einer Seite, das Keyword auch prominent vorkommt.

Also besteht folgende Herausforderung: Man braucht alle Keywords, mit denen man rankt, sowie die zugehörigen URLs. Da eine URL für mehr als ein Keyword ranken kann bzw. das sogar die Regel ist, muss man noch entscheiden, welches Keyword man zur Optimierung haben möchte. Hier in diesem Skript wird das Keyword verwendet, das am häufigsten bei Suchen aufgetaucht ist (Anzahl Impressions in den Suchergebnissen). Diese Daten bekommt man kostenlos von der Google Search Console. Wie man sich per Datenschnittstelle (API) anbindet, konnten Sie bereits in der letzten Ausgabe Nr. 65 lesen und ausprobieren. Die URLs, die Google Ihnen zurückliefert, werden mit dem Screaming Frog gecrawlt, denn über dieses Tool kommt man am einfachsten an den Title, die Description und die Inhalte der H1. Diese gesammelten Daten werden dann in einem weiteren Schritt in R sortiert und gefiltert. Als Ergebnis bekommt man eine Datei mit allen Keywords und URLs, in denen das Keyword eben noch nicht im Title/Desc/H1 entsprechend vorkommt. Was macht man mit dieser Liste? Mit einem wachen und intelligenten Auge durchsehen und die URLs markieren, bei denen eine Änderung für das Keyword Sinn macht. Dann wird es entsprechend ergänzt. Und voilà, die Rankings dafür sollten merklich ansteigen …

Bitte beachten Sie, dass diesmal das Skript zweigeteilt ist, weil es durch die Beschaffung einer weiteren Datei durch

INFO

Wer R noch nicht installiert hat, kann das schnell und einfach bewerkstelligen. Unter cran.r-project.org/bin/ findet man die aktuellen Versionen für Windows (32/64 Bit), MacOS (X) und Linux. Dort im Unterverzeichnis „base“ liegen dann die installierbaren Dateien. Nach der Installation holt man sich am besten gleich unmittelbar die Software „R-Studio“, die sich als eine Art Hülle um den Kern von R als grafische Benutzeroberfläche legt und viele Tools und Eingabehilfen zur Verfügung stellt. Die Bedienung von R wird mit R-Studio fast zum Kinderspiel. Man findet sie unter: *rstudio.com/products/rstudio/download/*

TIPP

Wenn Sie die beiden R-Skripte in das linke obere Fenster von RStudio einkopieren, können Sie diese bequem Schritt für Schritt abarbeiten lassen. Dazu stellen Sie jeweils den aktiven Cursor auf die Befehlszeile oder markieren einen längeren Block mit Befehlen mit der Maus. Dieser wird dann farblich markiert, wie Abb. 2, Ziffer 1 beispielhaft zeigt. Anschließend lösen Sie den Befehl oder die markierten Befehle mit den Tasten STRG und Enter aus. Dabei wird dann tatsächlich nur der Codeteil aktiviert, der markiert wurde (bzw. die Zeile mit dem aktiven Cursor). So können Sie nach und nach alle Skriptschritte ausführen und beobachten, was diese im Consolenfenster links unten auslösen. Und Sie können den Ablauf exakt steuern und ggf. besser verstehen, was hier im Einzelnen passiert.

einen Vorgang mit dem Screaming Frog unterbrochen wird.

Sie können wie immer einfach die beiden Skripte mit dem Zwischenschritt zum Screaming Frog hintereinander auf einmal durchlaufen lassen oder step by step bzw. Befehl für Befehl (siehe Tippkasten).

Am Ende bekommen Sie eine kleine Übersichtsgrafik erzeugt, wie die Verhältnisse zwischen „Keyword vorhanden/nicht vorhanden" und Title, Description und H1 sich darstellen.

Zusätzlich wird eine CSV-Datei erzeugt, die alle für Daten eine weitere Bearbeitung, Sortierung und Filterung erlaubt und die Ergebnisse tatsächlich erst operativ „weiterverarbeitbar" im Sinne einer To-do-Liste macht. Dies ist ja ein oft beklagter Mangel, dass viele SEO-Tools zwar schöne Oberflächen aufbereiten, aber man keine vernünftige Liste erhält/downloaden kann, mit der man weiterarbeiten kann.

Schritt 1: Die Daten aus der Google Search Console holen und Parameter festlegen

Wie Sie sich mit R an die Search Console anbinden können, konnten Sie in der letzten Ausgabe der Website Boosting lesen bzw. nachvollziehen. Hier haben wir nun gleich eine weitere sinnvolle Anwendung für Sie. Sollten Sie die Anbindung noch nicht vollzogen haben (muss nur einmalig erledigt werden), sollten bzw. müssen Sie dies natürlich nachholen, bevor Sie das neue Skript hier nutzen können.

Nach dem Start von R bzw. RStudio werden wie immer erst die notwendigen Bibliotheken geladen. Sind diese noch nie verwendet worden, müssen Sie zunächst (einmalig) installiert werden. Das geht ganz einfach mit den Befehlen:

```
install.packages(„googleAuthR“)
install.packages(„searchConsoleR“)
install.packages(„tidyverse“)
install.packages(„janitor“)
```

Anschließend werden die Bibliotheken in den aktiven Arbeitsspeicher von R geladen:

```
library(googleAuthR)
library(searchConsoleR)
library(tidyverse)
library(janitor)
```

Wie aus der letzten Ausgabe schon bekannt, müssen noch die Rechte für die Search Console hinterlegt werden. In diesem Fall reichen natürlich Leserechte (readonly):

```
options(googleAuthR.scopes.
selected = „https://www.goog-
leapis.com/auth/webmasters.rea-
donly“)
```

Die zur Authentifizierung nötige Datei von Google haben Sie ebenfalls noch vom Beispiel in der letzten Ausgabe. Sollte diese noch im Arbeitsverzeichnis von R liegen, brauchen Sie hier nichts weiter zu machen. Der Befehl lautet:

```
PATH_TO_KEY_FILE <- „r-leuch-
tungen-cf82XXXXXX718.json“
```

Der Dateiname kann natürlich bei Ihnen anders lauten, hier war er „r-leuchtungen" und Google hatte zur Erinnerung noch einen längeren Zahlencode hinten angehängt. Die Endung lautet .json. Sollte diese Datei vom letzten Mal sich nicht im Arbeitsverzeichnis befinden oder Sie an dieser Stelle eine Fehlermeldung von R bekommen, kopieren Sie diese einfach in das aktive Arbeitsverzeichnis. Alternativ können Sie rechts unten in RStudio unter „Files" das entsprechende Verzeichnis suchen und – wichtig – es dann unter More (mit dem Zahnradsymbol) mit „Set as working directory" zum Arbeitsverzeichnis machen. Im Consolenfenster links daneben erscheint dann das ausgewählte Verzeichnis bzw. der Befehl dazu, wie z. B. > setwd(„~/R/Test1"). Hier war das Arbeitsverzeichnis /Test1.

Jetzt kann der Verzeichnisname an die Funktion zur Authentifizierung gegenüber der GSC-API übergeben werden:

```
gar_auth_service(PATH_TO_KEY_
FILE)
```

Anschließend legen Sie fest, über welchen Zeitraum Sie die Daten aus der Search Console haben möchten, indem Sie in den beiden folgenden Zeilen Ihre individuellen Daten eingeben bzw. die Vorgaben überschreiben. Meist empfiehlt es sich, ein Jahr zurück zu nehmen. Beachten Sie aber bitte, dass Google für die vergangenen drei Tage keine GSC-Daten via API zur Verfügung stellt. Wenn Sie einen Tag aus der vergangenen Woche nehmen, liegen Sie meist gut damit. Natürlich kann man auch Daten für einen einzelnen Tag holen. Soll das der heutige Tag sein, müssen Sie allerdings wie erwähnt mindestens drei Tage damit warten.

```
START_DATE <- „2020-01-01“
END_DATE <- „2020-12-31“
```

In der folgenden Zeile tragen Sie die genaue Domain-URL ein, mit der die Domain in der Search Console geführt wird, bzw. überschreiben *„www.website-boosting.com/"* mit Ihrem Domainnamen.

```
GSC_PROP <- „https://www.websi-
teboosting.com/“
```

Nachdem fast jede Domain für ihren eigenen Namen in unterschiedlichen Schreibweisen bestens rankt, macht es keinen Sinn, diese Begriffe mit abzufragen bzw. Keyword-Kombinationen, die diese Worte enthalten. Daher haben Sie im nächsten Codeabschnitt die Möglichkeit, sog. „Brand"-Begriffe zu hinterlegen. Im weiteren Verlauf werden alle Rankings mit diesen Begriffen ausgenommen.

```
BRAND <- c(„website boosting“,
           „websiteboosting“)
```

Trennen Sie einfach verschiedene Schreibweisen mit einem Komma, wie im Beispiel gezeigt, und setzen Sie alle nötigen Kombinationen in Anführungszeichen. Würde man z. B. eine weitere Ausnahme definieren wollen oder typische Fehlschreibungen (hier mit drei „o“), dann würden die jetzt vier Zeilen wie folgt lauten:

```
BRAND <- c(„website boosting“,
           „websiteboosting“,
           „website booosting“,
           „websiteboosting“)
```

Den folgenden Schritt müssen Sie nicht verstehen. Der baut die eben angegeben Brandbegriffe so um, dass sie später bei der Abfrage in den hinterlegten Kombinationen ausgenommen werden.

```
BRAND_REGEX <- str_c(BRAND,
collapse = „|“) %>%
  str_c(„(„, ., „)“)
```

Jetzt geht es ans Eingemachte. Die Daten werden tatsächlich per API abgeholt. Dabei bildet das Skript zur Not mehrere Abfrageslots (Pages), die bei einer Domain mit vielen Keywords durchaus schnell nötig werden. Dieser Schritt kann etwas dauern! Bitte warten Sie bei einer Einzelzeilen-Abarbeitung ab, bis die Console die Meldung ausgibt, dass alles „downgeloadet“ ist.

```
gsc_data <- search_analytics(
   siteURL = GSC_PROP,
   startDate = START_DATE,
   endDate = END_DATE,
   dimensions = c(„page“,
   „query“),
   searchType = „web“,
   walk_data = „byBatch“,
   rowLimit = 50000)
```

In diesem Beispiel stünde dann im Consolenfenster, dass es drei API-Aufrufe (Calls) braucht, die in einzelnen „Pages“ abgerufen werden:

```
With rowLimit set to 50000 will
need up to [2] API calls

Page [1] of max [2] API calls
Downloaded 25000 rows

Page [2] of max [2] API calls
Downloaded 25000 rows
```

TIPP

Bitte beachten Sie: Manuell bekommen Sie per Download in der Google Search Console immer nur 1.000 Datensätze (Zeilen) übertragen. Tatsächlich kennt Google jedoch selbst bei kleinen Domains deutlich mehr Keywords, für die die Domain in den Suchergebnissen angezeigt wurde. Das R-Skript in dieser (und der letzten) Ausgabe holt jedoch tatsächlich alle Keywords für Sie ab. Dies sollten Sie im Kopf behalten, wenn Sie der Meinung sind, das könne man auch alles manuell per Download machen und müsse diesen Umweg über die API nicht gehen. Das wäre eine krasse Fehleinschätzung. Sie werden ggf. erstaunt sein, für welche Begriffe Ihre Domain tatsächlich am Ende bisher gerankt hat! Insofern lohnt sich der Aufwand für ein wenig Einarbeitung durchaus in den allermeisten Fällen. Insbesondere auch, weil Sie über die API die Kombination aus rankender URL und zugehörigen Suchbegriffen bekommen. Diese Information steht über die Webseite der Search Console nicht zum Export bereit!

Das Skript ist so eingestellt, dass es 50.000 Zeilen über die API abfragt. Sollte Ihre Website für mehr URL- / Suchanfragen-Kombinationen ranken, können Sie das Limit über das Argument „rowLimit“ erhöhen. Hier müssen Sie eventuell einfach ein wenig herumprobieren, welches Limit für Sie passend ist.

Anschließend werden die Daten bereinigt und wie oben beschrieben gefiltert und sortiert.

```
top_query_page <- gsc_data %>%
   filter(!str_detect(page,
   „#.*“)) %>%
   filter(!str_detect(query,
   BRAND_REGEX)) %>%
   group_by(page) %>%
   slice_max(impressions, n =
   1) %>%
  ungroup()
```

Danach wird eine Liste mit den URLs exportiert, damit sie mit dem Screaming Frog als Liste gecrawlt werden kann:

```
top_query_page %>%
  select(page) %>%
  write.table(„pages.txt“, quote
  = FALSE, row.names = FALSE,
  col.names = FALSE)
```

Im Arbeitsverzeichnis von R steht nun eine neue Datei namens „pages.txt“, deren Inhalte für den nächsten Schritt benötigt werden.

Schritt 2: Den Screaming Frog laufen lassen

Der nächste Schritt ist recht einfach und schnell beschrieben und in Abbildung 3 visualisiert. Öffnen Sie den Screaming Frog und stellen Sie zunächst den Arbeitsmodus über „Mode“ auf „List“ um (Ziffer 1). Das bewirkt, dass das lange Eingabefeld oben verschwindet, in das man normalerweise die Startadresse für den Crawler eingibt. Der „List“-Modus erlaubt es, dem Frog gezielt einzelne URLs vorzugeben, die er dann crawlt.

Öffnen Sie dann die Testdatei „pages.txt“ (Ziffer 2) und drücken Sie die Tastenkombination STRG und A. Das markiert den kompletten Text der Datei. Sie können natürlich auch alternative Methoden zur Textmarkierung verwenden. Anschließend kopieren Sie den markierten Text in die Zwischenablage (Ziffer 3), bei Windows z. B. über die Tastenkombination STRG und C. Die Datei „pages.txt“ können Sie jetzt bereits wieder schließen. Sie hat ihre Schuldigkeit getan.

Klicken Sie jetzt im Screaming Frog auf „Upload“ (Ziffer 4) und wählen Sie „Paste“ zum automatischen Einfügen aus. Alternativ könnte man natürlich auch per Verweis auf das gespeicherte File über „from a file“ zugreifen. Der Frog liest jetzt alle URLs aus der Zwischenablage ein und er muss dann nur noch gestartet werden. Jetzt werden alle

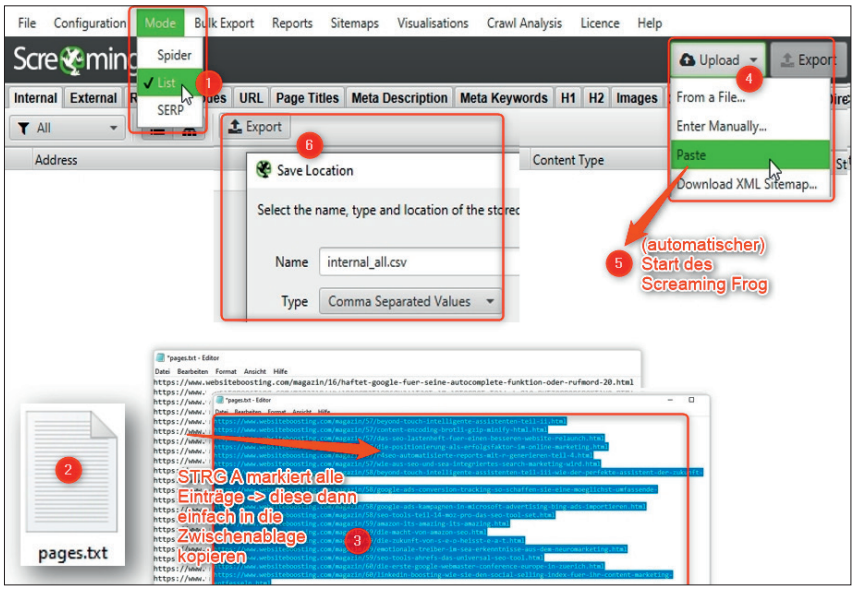


Abb. 3: Der prinzipielle Ablauf beim Zwischenschritt über den Screaming Frog

TIPP

Wenn Sie noch keine Vollversion des Screaming Frog besitzen, können Sie ggf. auch mit der Testversion arbeiten. Diese kann allerdings nur bis max. 500 URLs verarbeiten. Teilen Sie in diesem Fall die URLs in mehrere Teile auf und starten Sie mehrere Durchläufe nacheinander. Die erzeugten Dateien benennen Sie dann einfach jeweils vor dem Speichern um und kopieren am Ende die einzelnen Inhalte in eine einzige Datei namens „internal_all.csv“. Diese kann dann über den zweiten Teil des R-Skriptes weiterverarbeitet werden.

hinterlegten URLs aufgerufen und die Strukturdaten jeder URL ermittelt, u. a. eben auch Titel, Desc. und H1. Ist der Crawlvorgang beendet (ersichtlich rechts unten an der 100 %-Angabe), brauchen Sie nur noch über den Button „Export“ (Ziffer 6) eine CSV-Datei mit dem vorgeschlagenen Namen „internal_all.csv“ erzeugen bzw. diese am besten gleich direkt in das Arbeitsverzeichnis von R kopieren, wo auch die „pages.txt“ und die Authentifizierungsdatei liegen. Auf diese Datei greift dann der zweite Teil des R-Skripts zu, daher muss der Dateiname auch wirklich stimmen.

Schritt 3: Die fertige Auswertung erzeugen

Jetzt ist alles Manuelle geschafft und der Rest des Skripts bzw. der zweite Teil kann automatisch und komplett durchlaufen.

```
crawl <- read_csv(„internal_all.csv“) %>%
  clean_names() %>% # Spaltennamen normalisieren
  select(address, status_code,
    title_1, meta_descrip
    tion_1, h1_1)
```

Diese Zeilen lesen die relevanten Spalten aus dem Crawl bzw. die soeben erzeugten der Datei des Screaming Frog wieder ein. Damit das problemlos klappt, kopieren Sie ihn in das Arbeitsverzeichnis – sofern Sie ihn nicht sowieso schon vom Frog aus dorthin gespeichert haben.

Der nächste Codeblock sorgt dafür, dass beide Datenquellen vereint werden (Search-Console- und Frog-Daten) und einige Manipulationen vorgenommen werden. So werden u. a. URLs entfernt, die ggf. nicht mehr existieren. Wir arbeiten ja mit historischen Daten (z. B. ein Jahr) und da kann so etwas natürlich vorkommen. Nur wenn der Status 200 eingetragen ist, wird die URL weiterverwendet. Danach wird auf Kleinschreibung umgestellt („normalisiert“), Symbole durch Leerzeichen ersetzt und überflüssige Leerzeichen wieder entfernt. Danach wird geprüft, ob das Keyword (Query) im Title, der Description und/oder der H1 enthalten ist, und einige Werte werden einheitlich auf drei Nachkommastellen gerundet.

```
abgleich <- top_query_page %>%
  inner_join(crawl, by = c(„page“ = „address“)) %>%
  filter(status_code == 200) %>%
  select(-status_code) %>%
  mutate(across(c(title_1, meta_description_1,
    h1_1), tolower),
    across(c(title_1, meta_  description_1, h1_1),
      ~str_replace_all(.x, „[:punct:]“, „ „)),
    across(c(title_1, meta_description_1, h1_1),
      str_squish),
    across(c(title_1, meta_description_1, h1_1),
      ~case_when(str_detect(.x, query) ~ „ent-
      halten“, TRUE ~ „nicht enthalten“), .names =
      „query_in_{.col}“)) %>%
  mutate(across(c(ctr, position), ~round(.x, 3)))
    %>%
  arrange(query_in_title_1, query_in_meta_
    description_1, query_in_h1_1)
```

Das war bereits alles, was an Berechnungen durchgeführt werden muss. Der nächste Codeblock erzeugt dann schon eine kleine optische Übersicht bzw. „plottet“ alle Werte in eine Balkengrafik, die rechts unten in RStudio erscheint (im Reiter „Plots“). Abbildung 4 zeigt eine solche Grafik beispielhaft. Dabei sind jeweils die Anteile der URLs in rot markiert, die im Title, der Description oder der H1 das Keyword mit den meisten Impressions für die jeweilige URL nicht enthalten.

```
abgleich %>%
  select(starts_with(„query_in_“)) %>%
  pivot_longer(everything()) %>%
  ggplot(aes(name, fill = value)) +
  geom_bar(position = „fill“) +
  labs(title = „Anteile der enthaltenen Queries“,
       x = NULL,
       y = NULL,
```

```
      fill = „Query enthalten?“) +
  scale_y_continuous(labels = scales::percent_for-
mat())
```

Die Grafik dient natürlich nur zur schnellen Übersicht. Wertvoll wird die ganze Berechnung erst mit der Ausgabe einer weiterverarbeitbaren Datei. Dies wird mit einem einzigen Befehl erledigt:

```
write_excel_csv2(abgleich, „abgleich.csv“, na = „“)
```

Nach der Ausführung dieser Befehlszeile finden Sie im Arbeitsverzeichnis von R die CSV-Datei „abgleich.csv“. Dort finden Sie alle URLs mit dem Keyword, das am meisten Impressions erzeugt hat, und neben den Inhalten von Title, Desc und H1 auch, ob das Keyword dort vorkommt. Abbildung 5 zeigt den Inhalt dieser Datei – allerdings optisch schon etwas aufbereitet. Die Bereiche mit den Ziffern 1–3 zeigen mit bedingter farbiger Formatierung in Grün oder Orange an, ob das Keyword in der Spalte „query“ in den drei Bereichen enthalten ist oder nicht.

Was macht man am Ende mit der so erzeugten Datei?

Bevor man diese Daten als Arbeitsliste für Optimierungsmaßnahmen verwendet, kann und sollte man vielleicht noch etwas filtern und sortieren – der Effizienz wegen. Geht es z. B. darum, die sog. „Low Hanging Fruits“, also die tief hängenden Früchte zu ernten, markiert bzw. filtert man die Spalte „position“ z. B. auf Werte zwischen 11 und 39. Das bedeutet, die Rankings waren (zumindest grob) auf den Suchergebnisseiten zwei, drei und vier. Von dort kommt aber bekanntlich kein oder nur wenig Traffic. Da Google aber die jeweilige URL mit diesem Keyword offenbar schon sehr spannend findet – aber eben noch nicht gut/passend genug für Seite eins, kann man mit der Title- und H1-Optimierung hier sehr leicht nachhelfen. So passt man die Texte dort entsprechend an und setzt das Keyword

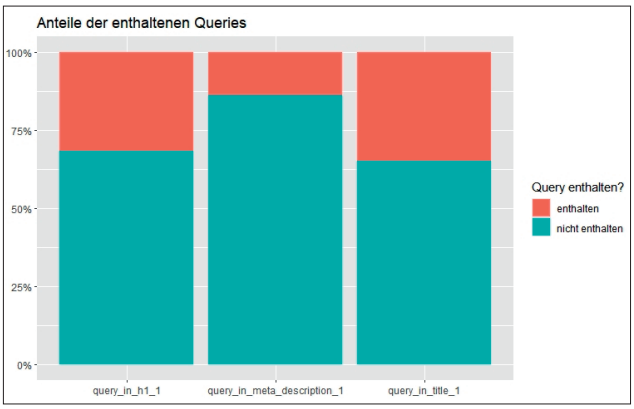


Abb. 4: Das Skript erzeugt eine kleine Übersichtsgrafik für H1, Desc. und Title.

entsprechend mit ein. Nicht selten reicht das schon, um ein paar Rankingpositionen gutzumachen und damit auf die erste Seite zu springen. Je weniger stark ein Keyword „umkämpft“ ist, desto leichter kommt man mit dieser einfachen Methode nach vorne. Im tiefen Wasser, also bei sehr werthaltigen Keywords mit vielen Mio. Treffern, wird die Vorwärtsbewegung natürlich in der Regel entsprechend geringer sein. Sie müssen natürlich auch im Kopf behalten, dass Sie nicht aus Versehen Keywords im Title/H1 austauschen, die für Sie wichtig sind und die vielleicht schon auf Seite 1 ranken!

Fazit:

Nirgends gilt so wie bei der Suchmaschinenoptimierung: A fool with a tool is still a fool. Insofern muss man jede Änderung auch immer mit wachem Auge auf die Sinnhaftigkeit prüfen. Ebenso natürlich, ob das jeweils betrachtete Keyword auch wirklich inhaltlich gut zur URL und auch generell gut zur Intention der Site passt.

Ein rein mechanisches „für alles ranken wollen“ wäre hier extrem kontraproduktiv. Das Skript hier kann Ihnen nur helfen, mögliche Potenziale aufzuzeigen und aus der oft schieren Menge an Daten eine schnelle Vorselektion zu ermöglichen. Was Sie daraus machen, müssen und sollten Sie selbst entscheiden. ¶

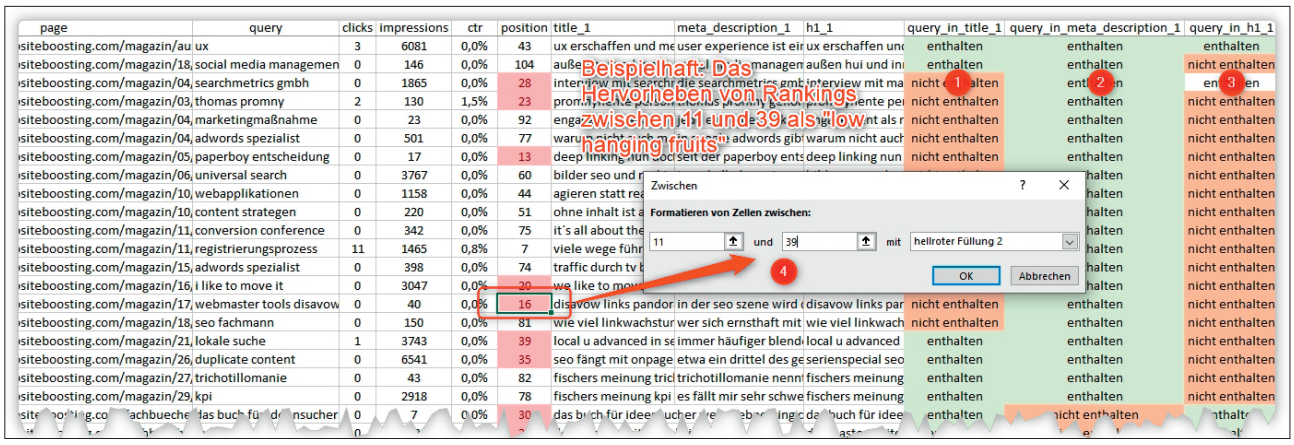


Abb. 5: In der am Ende erzeugten Datei finden Sie alles zur weiteren Bearbeitung

CODE

```
library(searchConsoleR)
library(tidyverse)
library(janitor)

# Authentifizierung --------

options(googleAuthR.scope

PATH_TO_KEY_FILE <- "r-1

gar_auth_service(PATH_T

#### TOEDIT: Berichtszei
START_DATE <- "2019-11-0
END_DATE <- "2020-11-30"

#### TOEDIT: Abzufra
# list_websites() %>% View()
GSC_PROP <- "htt

#### TOEDIT: Begriffe eintragen, der den eigenen Brand ausschlieÃt
BRAND <- c("website boosting",
           "websiteboosting")

# Begriffe zusammenbauen, um die Brand-Phrasen spaeter auszuschliessen
BRAND_REGEX <- str_c(BRAND, collapse = "|") %>%
  str_c("(", ., ")")

# Daten von der API abfrgen
gsc_data <- search_analytics(siteURL = GSC_PROP,
                             startDate = START_DATE,
                             endDate = END_DATE,
                             dimensions = c("page", "quer
                             searchType = "web",
                             walk_data = "byBatch",
                             rowLimit = 50000)

# Top Keywords pro Seite ermittlen
top_query_page <- gsc_data %>%
```

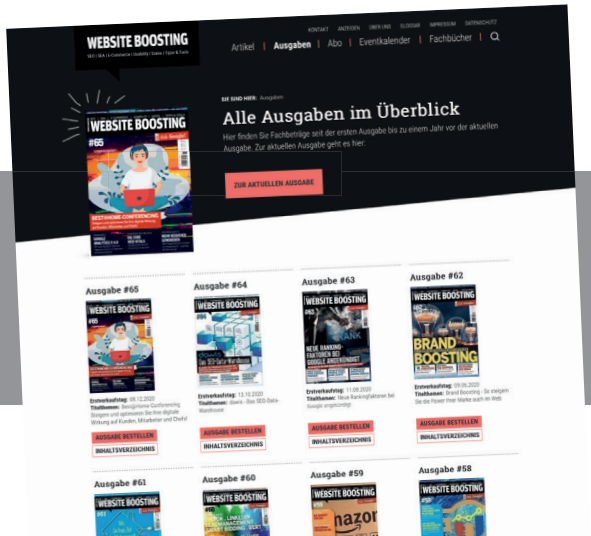
```
#### TOEDIT: Berichts
START_DATE <- "2019-11-0
END_DATE <- "2020-11-30"

#### TOEDIT: Abzufragende Prop
# list_websites() %>% View()
GSC_PROP <- "https://www.website

#### TOEDIT: Begriffe eintragen,
BRAND <- c("website boosting",
           "websiteboosting")

# Begriffe zusammenbauen, um di
BRAND_REGEX <- str_c(BRAND, co
  str_c("(", ., ")")

# Daten von der API abfr
sc_data <- search_a
```

TIPP

Wichtiger Hinweis zum Skript

Wenn Sie das Skript nicht einzeln wie hier beschrieben abarbeiten, sondern einfach in einem Stück laufen lassen möchten, müssen Sie unbedingt die Parameterdaten bei den Markern „#### TOEDIT“ einsetzen bzw. die Vorgaben überschreiben.

Den kompletten Code zum direkten Einkopieren in R erhalten Sie unter » http://einfach.st/rcode6



Wer Lust bekommen hat, tiefer in R einzusteigen, dem sei die fünfteilige Serie von Patrick Lürwer in der Website Boosting, Ausgaben 54 bis 58 empfohlen. Diese sind bereits frei online und als PDF-Version verfügbar. In den Ausgaben 52 und 53 finden Sie bei Bedarf noch weitere Tipps und Anleitungen für den Screaming Frog. Zur Auswahl für die kostenlosen Ausgaben (derzeit Ausgabe 1- 60) kommen Sie am besten über *www.websiteboosting.com/magazin.html*.