Mario Fischer, Patrick Lürwer

R-LEUCHTUNGEN!

Teil 1: Welche Ihrer Seiten bringen eigentlich den meisten Suchtraffic?

In den letzten Ausgaben der Website Boosting konnten Sie in einer Serie von Patrick Lürwer nachvollziehen, wie man die kostenlose Software R verwendet, was sie leistet und wie man sie nutzbringend für die eigene Arbeit 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, starten wir ab dieser Ausgabe eine neue anwendungsorientierte Serie. 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. Die nachfolgende Auswertung kann man auch mit reinem Copy & Paste erzeugen – und das in nur wenigen Minuten.

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.

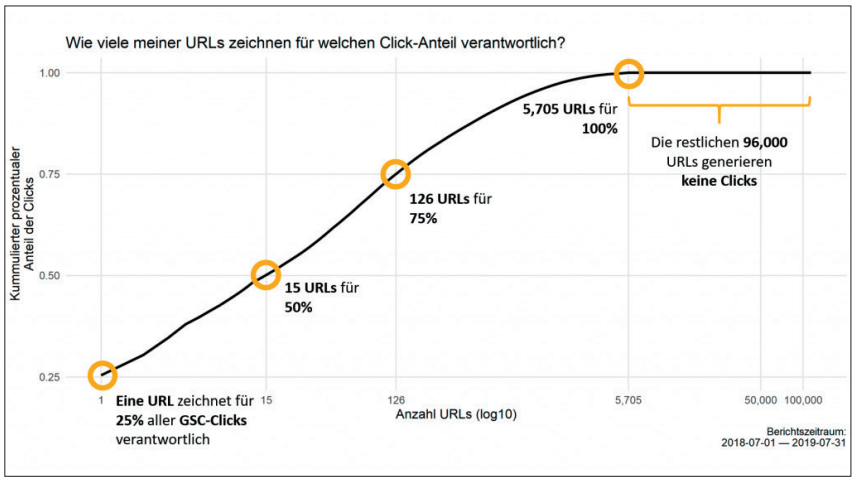


Abb. 1: Das Objekt der begründeten Wissbegierde – wie viele URLs sind eigentlich für die Klicks verantwortlich?

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/*

Der Fall bzw. das Problem: Was wird geklickt – was kann weg?

Viele Sites schleppen vergleichsweise viel Seitenballast mit sich herum, um den sich oft niemand mehr so richtig kümmert. Spätestens bei einem Relaunch stellt dann hoffentlich jemand die Frage, ob man wirklich alle URLs bzw. Webseiten mit in die neue Umgebung umziehen möchte. Der Aufwand dafür kann ja durchaus nicht unerheblich sein bzw. auch unnötig Geld und – wichtiger – unnötig Zeit kosten, von der man beim Relaunch ja bekanntlich nie genug hat.

Natürlich kann man nun einfach nach Bauchgefühl jede einzelne Seite durchgehen und prüfen, für welche Suchbegriffe diese auf welchen Positionen rankt, und auch, wie viel Traffic über Google da denn tatsächlich kommt. Viel ökonomischer ist es aber, sich erst einmal rein datengetrieben einen Überblick zu verschaffen.

Kennzeichnend für R sind die vielen Bibliotheken/Packages, die den Funktionsumfang teils enorm erweitern und an denen eine große Community freiwillig arbeitet. Ein solches Package muss man nur einmalig installieren, danach kann es mit seinem Namen einfach aufgerufen werden, falls man es erneut braucht.

Für die vorliegende Aufgabe braucht man die folgenden Packages:

```
janitor
tidyverse
```

Falls diese bisher noch nicht verwendet wurden, installiert man sie ganz einfach über R-Studio unter "Tools/Install Packages". In das aufpoppende Menü trägt man den Namen des Packages ein, z. B. googleAuthR, das dafür sorgt, dass man sich mit R über die Google-API Daten aus Google-Diensten holen kann. Das Package „searchConsoleR" sorgt dann beispielsweise dafür, dass R die Daten direkt anfordern kann.

Sind die beiden Packages einmal installiert, braucht man diesen Schritt wie erwähnt nicht mehr durchlaufen.

TIPP

Wer sich bereits mit R ein wenig auskennt, kann mit den Packages „googleAuthR" und „searchConsoleR" die Daten auch direkt aus der Search Console per API holen und mit dem Datensatz der URLs verknüpfen. Für Einsteiger ist es sicherlich zunächst deutlich leichter, einfach die API-Funktion des Screaming Frog zu nutzen und sich die Klick-Daten gleich direkt ein einem Rutsch mit abzuholen.

Möchte man bei einer R-Sitzung ein oder mehrere Packages nutzen, ruft man die bereits installierte Bibliothek ganz einfach mit dem Befehl „library(hier_den_Namen)" auf. Im vorliegenden Fall also mit:

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

Schritt 1: Die Daten besorgen

Zunächst braucht man eine Liste der URLs einer Domain und die Daten aus der Search Console für jede URL. Die bekommt man am einfachsten über einen Crawler wie z. B. den Screaming Frog. Das Vorgehen ist dabei relativ simpel. Im Fall des Screaming Frog verbindet man sich über Configuration/API-Access – Google Search Console. In dem erscheinenden Fenster wählt man den entsprechenden Google-Account aus, der Zugriff auf die Search Console erlaubt. Unter der Reiter „Date Range" wählt man anschließend noch aus, von wann bis wann man die Daten aus der Search Console haben möchte. Dabei muss man berücksichtigen, dass dort nur jeweils die letzten 16 Monate gespeichert werden.

Danach gibt man den entsprechenden Domainnamen oben in die Adresszeile des Screaming Frog ein und

INFO

Damit Sie den Code nicht einzeln abtippen müssen, können Sie ihn unter *http://einfach.st/rcode2* als Textdatei downloaden und sich entweder alles auf einmal oder auch Befehl für Befehl herauskopieren.

klickt auf Start. Wann der Crawl fertig durchgelaufen ist, erkennt man rechts unten in der Fußzeile des Programms. Steht dort 100 %, ist alles beendet. Kontrollieren Sie nun, ob die Search Consoledaten übergeben wurden. Dazu einfach in dem Hauptfenster mit den URLs und Kenndaten ganz nach rechts scrollen und den Reiter „Search Console" auswählen. Die vorletzten Spalten sind die Clicks, Impressions, CTR und die Position pro URL.

Ein Klick auf den Button „Export" unter den Reitern oben öffnet den Datei-speichern-Dialog. Geben Sie dem File beim Speichern den Namen **„search_console_all.csv",** weil der Code später genau auf diesen Namen zugreift. Selbstverständlich kann man auch jeden anderen Namen wählen, muss dann aber weiter unten im Code natürlich den vorgegebenen Filenamen durch den selbst vergebenen ersetzen.

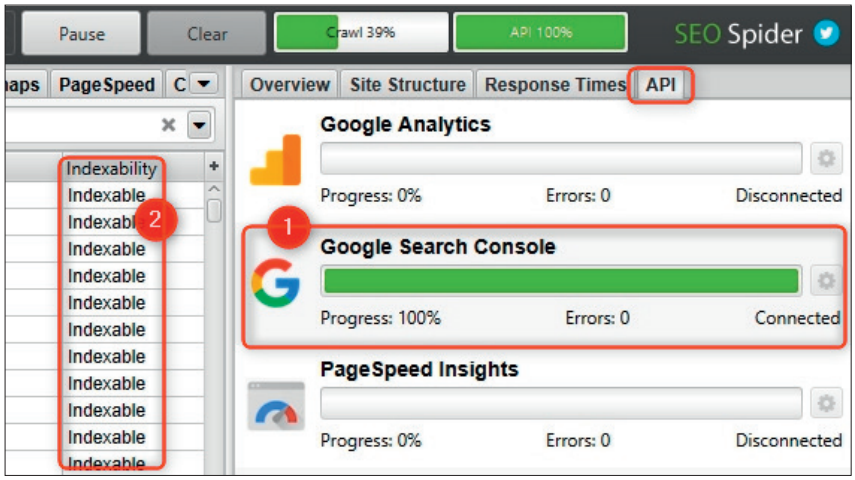


Abb. 2: Über den Reiter „API" kann man den Datenabruf kontrollieren (Ziffer 1)

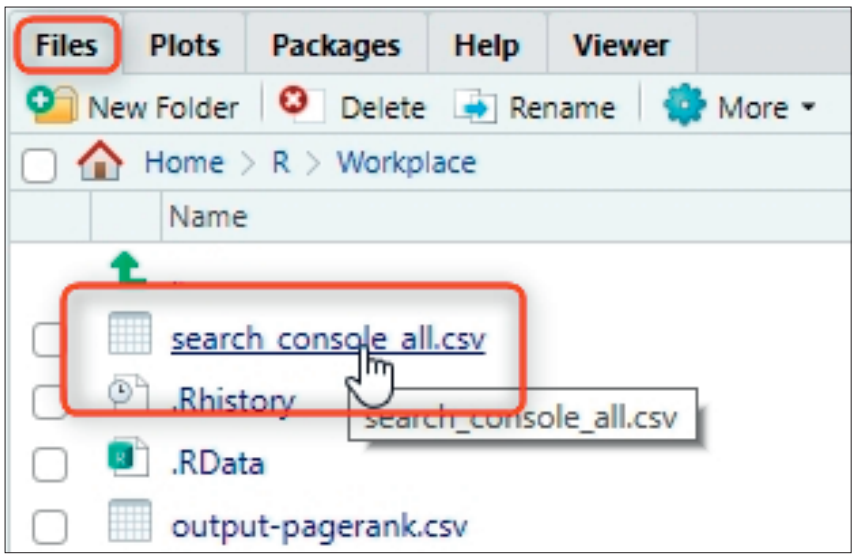


Abb. 3: Kontrollieren Sie, ob Ihr File auch im Workplace von R liegt

Ab in R und los gehts

Am besten speichert man das CSV-File gleich direkt in das Arbeitsverzeichnis von R. Das finden Sie bei einer Standardinstallation im Verzeichnis von R und dort unter „Workplace". Noch einfacher ist es aber, sich das Arbeitsverzeichnis einfach in R mit dem Befehl

```
getwd()
```

anzeigen zu lassen. Getwd steht dabei als Abkürzung für get working directory. Wenn man das weiß, kann man es sich leichter merken. Im rechten unteren von den vier Fenstern in R-Studio können Sie unter dem Reiter „Files" dann nach dem Speichern auch das eben mit Screaming Frog erzeugte File erkennen.

Wenn die beiden Packages schon installiert sind, können diese nun direkt beim blinkenden Cursor aufgerufen bzw. aktiviert werden, wie oben bereits beschrieben wurde. Es schadet aber auch nichts, wenn man sie mehrfach aufruft bzw. aktiviert.

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

Jetzt kann man schon mit dem Umformen der Daten beginnen. Die Befehlszeile

```
crawl <- read_csv(„search_console_
all.csv") %>%
  clean_names()
```

sorgt dafür, dass die Namen der Spalten in den Daten für den einfacheren Umgang normalisiert werden.

Ein kleines Problem stellt der folgende Sachverhalt dar: Nicht zu jeder URL gibt es auch tatsächlich Klicks, denn einige URLs ranken mitunter gar nicht. Für solche URLs gibt die API überhaupt keinen Wert zurück und im Datenfeld steht dann „NA" für „not available", also nicht verfügbar. Das kann unterschiedliche Gründe haben. Aber mit nichts kann man nicht vernünftig rechnen, daher füllt man alle leeren Zellen bei den Klickwerten mit 0 auf.

```
crawl <- crawl %>%
  replace_na(list(clicks = 0))
```

Da in der Regel nicht alle URLs auch echten Dokumenten entsprechen, z. B. durch Weiterleitungen etc., filtert man anschließend nur die heraus, die in der Spalte „Indexability" den Wert „Indexable" stehen haben. Alle anderen werden ignoriert (siehe Abbildung 2 Ziffer 2).

```
crawl <- crawl %>%
    filter(indexability == „Indexable“)
```

Jetzt fehlen noch eine wichtige Sortierung und die Berechnung einiger Kennwerte für die spätere Darstellung. Zunächst werden die Zeilen absteigend nach Klicks sortiert. Anschließend wird der prozentuale Anteil jeder URL an den Gesamt-Klicks berechnet. Im nächsten Schritt werden die soeben berechneten Prozentwerte aufsummiert, damit später der Anteil in absteigender Reihenfolge gut ersichtlich ist. Anschließend wird noch die Metrik „url_rank" angelegt, nach der die x-Achse aufgebaut wird.

```
crawl <- crawl %>%
    arrange(desc(clicks)) %>%
        mutate(pct_clicks = clicks / sum(clicks),
            cum_pct_clicks = cumsum(pct_clicks),
            url_rank = row_number())
```

Das waren auch schon alle nötigen Befehlszeilen zur Datenaufbereitung.

Die Grafik erzeugen

Für die grafische Ausgabe aller Klickdaten fügt man final noch die folgenden Befehlszeilen ein, die im Wesentlichen zur Gestaltung des Plots bzw. der Abbildung dienen.

```
ggplot(crawl, aes(url_rank, cum_pct_clicks)) +
    geom_line(size = 1) +
    theme_minimal() +
    theme(panel.grid.minor = element_blank()) +
    labs(title = „Wie viele der URLs zeichnen für welchen
Klick-Anteil verantwortlich?“,
        x = „Anzahl URLs“,
        y = „Kumulierter prozentualer\nAnteil der Klicks“)
```

Den Text für die Überschrift und die Achsenbezeichnung kann man natürlich nach eigenem Belieben anpassen. Noch ein letztes Mal „Return" drücken und das war es auch schon. Im rechten Grafikfenster (Plots) erscheint die fertige Abbildung. Je nach Rechnerpower dauert das zwischen einer Sekunde und nur wenigen mehr.

Wenn das URL-File aus dem Screaming Frog im Workingspace von R vorliegt, vergehen zwischen dem Start von

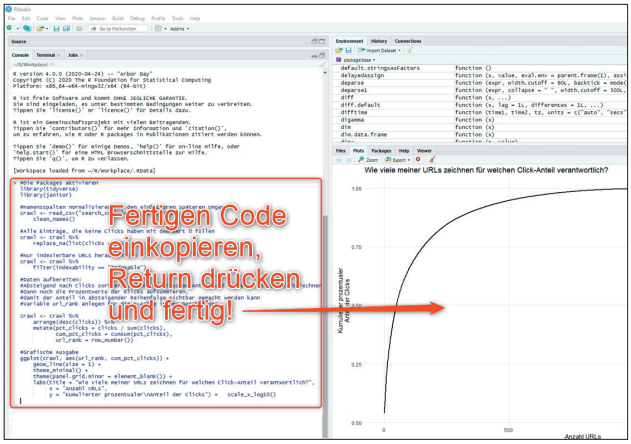


Abb. 4: Es ist wirklich so einfach – Code einkopieren, Return, fertig

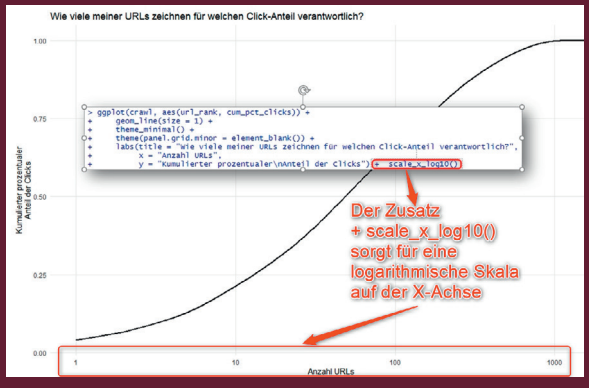
R-Studio, dem Einkopieren des gesamten Codes (downloadbar, siehe Kasten) am Stück und einem Return nur fünf bis zehn Sekunden. Selbstverständlich kann man auch über Excel alles einzeln und manuell so umformen, dass eine derartige Auswertung bzw. Abbildung erzeugt werden kann. Je nach Excelkenntnissen kann dabei aber durchaus auch schon mal eine Stunde vergehen. Und: Diese Arbeit muss man jedes Mal erneut machen!

Wie ist die Abbildung zu lesen?

Auf der y-Achse ist der summierte Anteil an Klicks zu sehen und die X-Achse stellt die dazu benötige Anzahl an URLs dar. Wie in Abbildung 5 ersichtlich ist, weist der Crawl insgesamt über 1.600 URLs auf. Allerdings haben nur 49 URLs für die Hälfte (Wert 0,5) des Traffics über die organische Suche bei Google gesorgt. 573 URLs haben tatsächlich

TIPP

Wenn die Kurve zu steil ist, weil tatsächlich nur sehr wenige URLs für den größten Teil verantwortlich sind, kann man die Skala auch relativ leicht mit einem Logarithmus skalieren. Dazu gibt man am Ende der Befehlskette nach der Zeile „y = „Kumulierter prozentualer\nAnteil der Clicks")" einfach ein Pluszeichen und einen Scale-Befehl ein bzw. fügt diesen hinten an. Die Erweiterung sieht dann so aus: **+ scale_x_log10()**



innerhalb eines Jahres nur ein Prozent des Traffics einsammeln können. Wie man diese genauen Werte ermitteln, wird gleich noch weiter unten erklärt.

So erschließt sich eine völlig neue Sichtweise auf die Leistung der eigenen Domain und man erhält harte Daten übersichtlich aufbereitet an die Hand.

Doch woher bekommt man die genauen Werte, wenn man bei der x-Achse nicht schätzen möchte? Mit ein paar weiteren Befehlen, die man einfach wieder in die Console einkopiert.

```
get_percentiles <- function(df, percentiles = c(0.25, 0.5, 0.75, 1)) {
    v <- integer()
    for (percentile in percentiles) {
        url_rank <- df %>%
        arrange(abs(cum_pct_clicks - percentile)) %>%
        .[[1, „url_rank“]]
        v <- c(v, url_rank)
    }
    names(v) <- percentiles
    return(v)
}
get_percentiles(crawl)
```

wirft in R die Zeilen

0.25	0.5	0.75	1
13	49	156	1389

aus. Die Lesart ist: 25 % aller Klicks kommen von 13 URLs, die Hälfte (0,5) von 49 URLs, drei Viertel (0,75) von 156 URLs und das restliche Quartil bis 100 % wird von 1.389 URLs erzeugt.

Tauscht man z. B. die letzte Zeile gegen die nachfolgende aus,

```
get_percentiles(crawl, c(.1, .5, .99))
```

werden individuell angepasste Percentile angezeigt. Im Beispiel wurden 10 %, 50 % und 99 % verwendet, was man natürlich selbst beliebig ändern kann.

Nun lässt sich am Ergebnis ablesen, dass 573 URLs tatsächlich nur das letzte eine Prozent auf 100 an Suchtraffic erzeugt haben. Rechnen Sie dazu 1.389 URLs (100 % der Klicks) – 816 URLs (die ersten 99 % der Klicks) = 573 URLs (das verbleibende eine Prozent der Klicks).

0.1	0.5	0.99
3	49	816

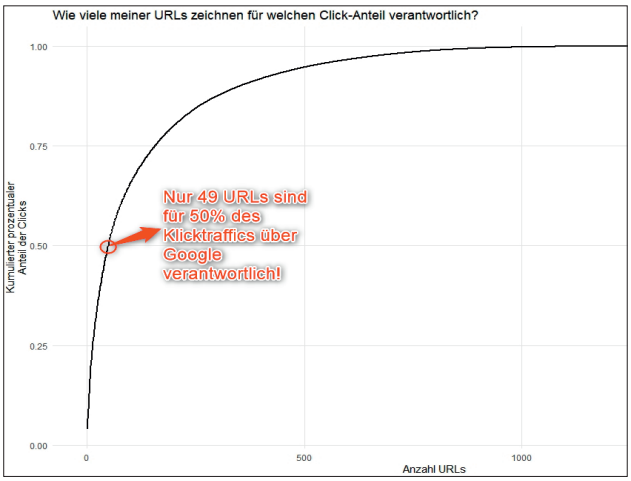


Abb. 5: Hier sieht man auf den ersten Blick, dass tatsächlich nur 49 URLs für die Hälfte des Google-Traffics aus dem letzten Jahr verantwortlich sind

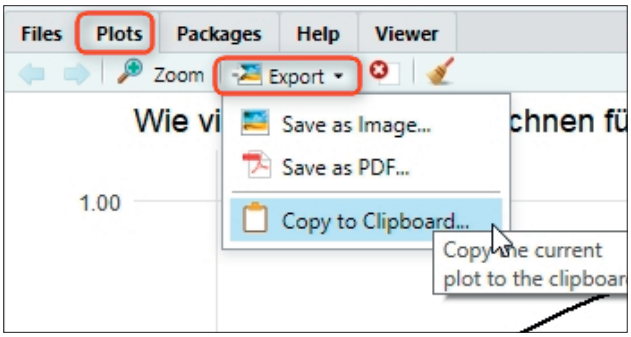


Abb. 6: Fertige Abbildungen lassen sich exportieren und weiterverwenden

Die erzeugte Abbildung selbst lässt sich im Reiter „Export“ unter „Plots“ exportieren oder in die Zwischenablage kopieren und z. B. in Powerpoint oder einem anderen Grafikprogramm weiter verschönern bzw. beschriften.

Kann ich das nicht auch in Excel?

Prinzipiell ja. Das kann aber durch die heftige Arbeit mit Formeln und Daten durchaus einige Zeit in Anspruch nehmen. Und auch hier braucht man schon deutlich fortgeschrittene Kenntnisse. In R erledigt man das mit dem vorgefertigten Code in nur wenigen Sekunden. Natürlich besteht auch die Möglichkeit, die Codezeilenreihenfolge abzuspeichern und dann nur mit einem einzigen Befehl abzurufen. Schneller ist wohl keine Software.

Das kleine Beispiel im ersten Teil dieser neuen Reihe zeigt recht deutlich, wo die Stärken von R liegen. In der Mächtigkeit im Datenumgang und dem Vorteils, einen Vorgang nur einmalig ausführen zu müssen, weil man ihn später immer wieder laufen lassen kann. Und im vorliegenden Fall reichen sogar Copy-&-Paste-Kenntnisse. Bei denen sollten Sie aber natürlich nicht stehen bleiben. Die Beispiele sollen Ihnen Appetit und Mut machen, Dinge einfach auch mal selbst auszuprobieren und danach dauerhaft davon zu profitieren. ¶

befehle - Editor

Datei Bearbeiten Format Ansicht Hilfe

```
# Codebeispiel für R - Website Boosting Ausgabe 62
# Alle Zeilen, die mit # beginnen, dienen nur der Erklärung
# und werden von R ignoriert.

# Hinweis. Die beiden Librarys müssen einmalig installiert
# werden. Wie das geht, steht im Heft!

# 1. Packages laden bzw. aktivieren

library(tidyverse)
library(janitor)

# 2. Namensspalten normalisieren für den einfacheren späteren Umgang
crawl <- read_csv("search_console_all.csv") %>%
  clean_names()

# 3. Alle Einträge, die keine Clicks haben mit dem Wert 0 füllen
crawl <- crawl %>%
  replace_na(list(clicks = 0))

# 4. Nur indexierbare URLs herausfiltern
crawl <- crawl %>%
  filter(indexability == "Indexable")

#´5. Daten aufbereiten:
  crawl <- crawl %>%
  arrange(desc(clicks)) %>%
    mutate(pct_clicks = clicks / sum(clicks),
    cum_pct_clicks = cumsum(pct_clicks),
    url_rank = row_number())

# 6. Grafische Ausgabe
  ggplot(crawl, aes(url_rank, cum_pct_clicks)) +
  geom_line(size = 1) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank()) +
  labs(title = "Wie viele meiner URLs zeichnen für welchen Click-Anteil verantwortlich?",
       x = "Anzahl URLs",
       y = "Kumulierter prozentualer\nAnteil der Clicks")

# ggf noch die nachfolgende Zeile an die letzte unter 6. anhängen für
# eine logarithmische Darstellung der X-Achse, das # dabei entfernen
#      +   scale_x_log10()

# 7. Percentile berechnen

get_percentiles <- function(df, percentiles = c(0.25, 0.5, 0.75, 1)) {
    v <- integer()
    for (percentile in percentiles) {
    url_rank <- df %>%
    arrange(abs(cum_pct_clicks - percentile)) %>%
    .[[1, "url_rank"]]
    v <- c(v, url_rank)
    }
```

Den kompletten Code zum direkten Einkopieren in R erhalten Sie unter » http://einfach.st/rcode2

TIPP

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- 54) kommen Sie am besten über *www.websiteboosting.com/magazin.html*.