Foto: lunewind / gettyimages.de

Patrick Lürwer

R4SEO: AUTOMATISIERTE REPORTS MIT R GENERIEREN (TEIL 3/4)

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.

Im vorherigen Teil dieser vierteiligen Artikelserie haben Sie gelernt, wie Sie fehlende Datumspunkte in Ihre DataFrames einfügen. Außerdem haben Sie die Berechnung des gleitenden Mittelwerts zur Glättung von Graphen angewendet und einen DataFrame transponiert. Damit bleiben noch zwei Aufgaben der Datenaufbereitung, bis Sie im nächsten Teil mit der Visualisierung beginnen können. Zum einen werden Sie in diesem Artikel die DataFrames mit den URLs resp. Queries aus der Google Search Console so aggregieren, dass Sie die Top-10-URLs bzw. -Queries nach Clicks in der Vorwoche erhalten. Zum anderen führen Sie die beiden DataFrames mit den Desktop- bzw. Mobile-Sichtbarkeitsdaten zusammen, um sie in einem Diagramm darstellen zu können.

```
189
190 # Daten aufräumen 02 ------------------------------------------------
191
192
193 # GSC ---------------------------------------------------------------
194
195 # Berechnung des Montags und Sonntags der Vorwoche
196 MONDAY_PREVIOUS_WEEK <- floor_date(today(), "week", 1) - 7
197 SUNDAY_PREVIOUS_WEEK <- MONDAY_PREVIOUS_WEEK + 6
198
199
200 # Page:Berechnung der Top 10 Seiten nach Clicks der Vorwoche
201 gsa_top_10_pages <- gsa_dim_date_page %>%
202
203   filter(between(date, MONDAY_PREVIOUS_WEEK, SUNDAY_PREVIOUS_WEEK)) %>%
204
205   group_by(page) %>%
206   summarise(sum_clicks = sum(clicks),
207             sum_impressions = sum(impressions),
208             ctr  = sum_clicks / sum_impressions,
209             weight_position = sum(impressions * position) / sum(impressions)) %>%
210   ungroup() %>%
211
212   top_n(n = 10,
213         wt = sum_clicks) %>%
214
215   arrange(desc(sum_clicks)) %>%
216
217   mutate(page = str_replace(page, GSC_PROP, "/"),
218          ctr = round(ctr, 3),
219          weight_position = round(weight_position, 2)) %>%
220
221   rename("Page" = page,
222          "Clicks" = sum_clicks,
223          "Impressions" = sum_impressions,
224          "CTR" = ctr,
225          "Position" = weight_position)
226
```

Abb.1: Berechnung der Top-10-Seiten nach Clicks der Vorwoche

```
1  # Hilfsfunktion
2  print_weekday <- function(date) {
3    print(paste(date, "->", weekdays(date)))
4  }
5
6
7  # Heutiges Datum
8  heute <- today()
9  print_weekday(heute)
10 # [1] "2019-01-20 -> Sonntag"
11
12
13 # Datum des Wochenbeginns
14 beginn_der_woche <- floor_date(today(), "week")
15 print_weekday(beginn_der_woche)
16 # [1] "2019-01-20 -> Sonntag" <- In den USA beginnt die Woche am Sonntag!
17
18
19 beginn_der_woche_montag <- floor_date(today(), "week", 1)
20 print(beginn_der_woche_montag)
21 # "2019-01-14 -> Montag" <- 1 = Wochenbeginn am Montag
22
23
24 # Beginn der Vorwoche
25 beginn_der_vorwoche <- floor_date(today(), "week", 1) - 7
26 print_weekday(beginn_der_vorwoche)
27 # "2019-01-07 -> Montag"
28
29
30 # Ende der Vorwoche
31 ende_der_vorwoche <- beginn_der_vorwoche + 6
32 print_weekday(ende_der_vorwoche)
33 # [1] "2019-01-13 -> Sonntag"
```

Abb.2: Die Berechnung des Montags und Sonntags der Vorwoche im Detail

Im letzten Artikel haben Sie die Daten in den DataFrames *gsa_dim_date* sowie *ga_sessions* transformiert. Damit fehlen noch die DataFrames für die URL- und Query-Performance-Daten (*gsa_dim_date_page, gsa_dim_date_query*) aus der Google Search Console (GSC) sowie den beiden DataFrames für die Desktop- resp. Mobil-Sichtbarkeitsdaten (*si_desktop, si_mobile*) aus SISTRIX.

Zum Start legen Sie wie bisher ein neues R-Skript an, benennen es mit *data_tidying_02.R* und kopieren den Inhalt des Skripts *data_tidying_01.R* hinein. Sie müssen dieses Mal keine neuen Packages installieren, sondern verwenden erneut jene, die Sie bereits aus dem vorherigen Artikel kennen. Führen Sie das neue Skript aus, damit wieder alle Daten zur Verfügung stehen.

Top-Seiten aus der Google Search Console ermitteln

Für den zu erstellenden Report wollen Sie zwei Tabellen generieren, die die Top-10-Seiten bzw. -Suchanfragen nach Clicks der Vorwoche darstellen. Die dafür benötigten Daten liegen jedoch aktuell noch auf Tagesbasis für den gesamten Berichtszeitraum in den DataFrames *gsa_dim_date_page* und *gsa_dim_date_query*.

Kopieren Sie daher den Code aus Abbildung 1 ans Ende des zuvor von Ihnen neu erstellten Skripts. Keine Sorge, Sie werden diese Monsterabfrage gleich Schritt für Schritt durchgehen, um im Detail zu verstehen, wie die Daten transformiert werden. Die Absätze zwischen den einzelnen Abschnitten sind nur der Lesbarkeit geschuldet und können von Ihnen weggelassen werden.

Sie möchten für Ihren Report die Vorwoche betrachten. Daher müssen Sie zunächst ermitteln, auf welches Datum der Montag bzw. der Sonntag gefallen ist. Dazu dienen die ersten beiden Code-Zeilen. Anhand dieser können Sie dynamisch, ausgehend vom aktuellen Datum, die beiden Datumspunkte berechnen. Dabei gibt es eine Besonderheit: In den USA beginnt die Woche am Sonntag. Wenn Sie mit Zeit-

TIPP

Um Ihnen das Abtippen des Codes zu ersparen, können Sie sich auch dieses Mal das Skript zu diesem Artikel unter folgendem Link auf GitHub herunterladen: *https://github.com/gettractiongmbh/r4seo_ws*. Ich empfehle Ihnen dennoch, den Code selbst zu schreiben, um ein Gefühl für Syntax und Funktionen zu bekommen.

reihen/-funktionen arbeiten, sollten Sie dies stets im Hinterkopf behalten. So wird bspw. auch in Google Analytics (GA) für die Diagrammdarstellung die amerikanische Woche genutzt. Wählen Sie im Kalender hingegen „Letzte Woche“, gilt wieder die deutsche Rechnung von Montag bis Sonntag.

Zurück zur Berechnung des Montags und Sonntags der Vorwoche. In Abbildung 2 sehen Sie die einzelnen Schritte im Detail. Die Funktion *floor_date()* aus dem Package *lubridate* kennen Sie bereits aus dem ersten Artikel. Dort haben Sie damit den Beginn des abzufragenden Berichtszeitraums berechnet. Mit dem Argument *week* wird der Funktion angezeigt, dass sie das Datum des Wochenbeginns, ausgehend vom heutigen Datum (*today()*), finden soll. Anschließend werden sieben Tage subtrahiert, um zum Montag der Vorwoche zu gelangen. Durch die Addition von sechs Tagen finden Sie dann den Sonntag. Die berechneten Datumspunkte schreiben Sie in die Variablen *MONDAY_PREVIOUS_WEEK* und *SUNDAY_PREVIOUS_WEEK*, um sie nachfolgend zum Filtern der DataFrames zu verwenden.

Im Anschluss führen Sie die Berechnung der Top-10-Seiten aus. Ihr DataFrame *gsa_dim_date_page* mit den Performance-Daten sieht wie in Abbildung 3 aus. Für jedes Datum des gesamten Berichtszeitraums, an dem die jeweilige URL in der Spalte page mindestens eine Impression generiert hat, liegen Ihnen die Metriken *clicks, impressions, ctr* und *position* vor.

Zunächst müssen Sie den DataFrame auf den Datumsbereich der Vorwoche filtern (Abbildung 1; 1). Dazu geben Sie *gsa_dim_date_page* an die Funktion *filter()*. Diese nimmt wiederum die Funktion *between()* auf. In dieser definieren Sie, dass die Spalte date dahingehend geprüft werden soll, ob die Werte darin zwischen den beiden Datumspunkten *MONDAY_PREVIOUS_WEEK* und *SUNDAY_PREVIOUS_WEEK* liegen. Übrig bleiben somit nur noch die Zeilen, welche Daten der Vorwoche enthalten.

Darauf folgt ein *group_by() (2)*, denn Sie wollen die Metriken für jede URL separat aggregieren. Wie die einzelnen Metriken aggregiert werden sollen, definieren Sie mit dem anschließenden *summarise()*. Die Operation ist etwas abstrakt, daher finden Sie in Abbildung 4 eine schematische Darstellung anhand von drei URLs. Der *group_by()* teilt den DataFrame in einzelne Untergruppen basierend auf den URLs auf. Dies erfolgt nur intern, d. h., würden Sie den Code nur bis zum *group_by()* ausführen, sähen Sie keinen Unterschied in der Darstellung zum initialen DataFrame.

Innerhalb des *summarise()* definieren Sie dann, welche neuen Spalten gebildet werden sollen und wie die Werte dieser Spalten berechnet werden. Für die Spalten clicks und impressions berechnen Sie die Summe (*sum()*) jeder URL-Gruppe und schreiben das Ergebnis in die Spalten *sum_clicks resp. sum_impressions*. Die *ctr* berechnen Sie hier neu, indem Sie die Werte der neuen Spalte *sum_clicks* durch *sum_impressions* teilen. Für die *position* können Sie nicht einfach den Mittelwert berechnen, da die Wertigkeit einer Ranking-Position von den Impressions abhängig ist. Sie können auf Ihrer Website eine Seite haben, die zu einer sehr seltenen Suchanfrage auf Position 1 rankt. Hat diese Seite jedoch

gsa_dim_date_page

Filter

	date	page	clicks	impressions	ctr	position
1	2018-09-19	https://dein-trueffel.de/trueffel-produkte/trueffelbutter/	4302	38018	0.11315693	4.285286
2	2018-09-20	https://dein-trueffel.de/trueffel-produkte/trueffelbutter/	787	6787	0.11595698	1.207603
3	2018-09-21	https://dein-trueffel.de/trueffel-produkte/trueffelbutter/	393	3207	0.12254443	1.489866
4	2018-09-23	https://dein-trueffel.de/trueffel-produkte/trueffelbutter/	263	1788	0.14709172	1.733781
5	2018-09-22	https://dein-trueffel.de/trueffel-produkte/trueffelbutter/	246	2058	0.11953353	1.681244
6	2018-10-27	https://dein-trueffel.de/trueffel-wissen/wo-wachsen-trueffel/	119	680	0.17500000	6.579412
7	2019-01-04	https://dein-trueffel.de/trueffel-wissen/wo-wachsen-trueffel/	119	574	0.20731707	4.750871

Abb.3: DataFrame gsa_dim_date_page

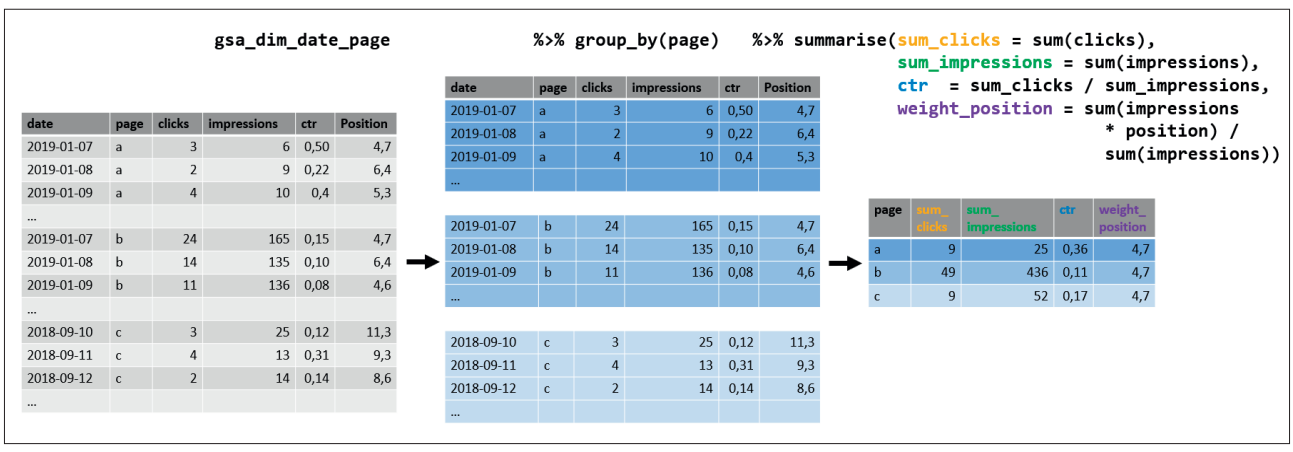

Abb.4: Schematische Darstellung von group_by() und summarise()

gsa_top_10_pages

Filter

	page	sum_clicks	sum_impressions	ctr	weight_position
1	https://dein-trueffel.de/	5	15	0.33333333	5.733333
2	https://dein-trueffel.de/trueffel-produkte/trueffelbutter/	107	1033	0.10358180	5.433688
3	https://dein-trueffel.de/trueffel-produkte/trueffelhonig/	4	6	0.66666667	5.666667
4	https://dein-trueffel.de/trueffel-produkte/trueffeloel/	2	20	0.10000000	18.150000
5	https://dein-trueffel.de/trueffel-rezepte/	2	14	0.14285714	14.571429
6	https://dein-trueffel.de/trueffel-rezepte/383-3/	4	44	0.09090909	36.909091
7	https://dein-trueffel.de/trueffel-rezepte/383-4/	17	143	0.11888112	8.580420
8	https://dein-trueffel.de/trueffel-wissen/arten/china-trueffel/	14	87	0.16091954	9.643678
9	https://dein-trueffel.de/trueffel-wissen/arten/fruehlingstrueffel/	2	6	0.33333333	6.833333
10	https://dein-trueffel.de/trueffel-wissen/arten/perigord-trueffel/	10	243	0.04115226	9.226337
11	https://dein-trueffel.de/trueffel-wissen/arten/wintertrueffel/	6	79	0.07594937	10.822785
12	https://dein-trueffel.de/trueffel-wissen/trueffelsuche/	48	576	0.08333333	8.373264
13	https://dein-trueffel.de/trueffel-wissen/trueffelsuche/trueffelhu...	4	125	0.03200000	13.056000
14	https://dein-trueffel.de/trueffel-wissen/trueffelsuche/trueffelsc...	4	273	0.01465201	7.736264
15	https://dein-trueffel.de/trueffel-wissen/wo-wachsen-trueffel/	212	1178	0.17996604	6.238540

Abb.5: DataFrame nach der Aggregation auf URL-Ebene

	page	sum_clicks	sum_impressions	ctr	weight_position
1	https://dein-trueffel.de/	5	15	0.33333333	5.733333
2	https://dein-trueffel.de/trueffel-produkte/trueffelbutter/	107	1033	0.10358180	5.433688
3	https://dein-trueffel.de/trueffel-produkte/trueffelhonig/	4	6	0.66666667	5.666667
4	https://dein-trueffel.de/trueffel-rezepte/383-3/	4	44	0.09090909	36.909091
5	https://dein-trueffel.de/trueffel-rezepte/383-4/	17	143	0.11888112	8.580420
6	https://dein-trueffel.de/trueffel-wissen/arten/china-trueffel/	14	87	0.16091954	9.643678
7	https://dein-trueffel.de/trueffel-wissen/arten/perigord-trueffel/	10	243	0.04115226	9.226337
8	https://dein-trueffel.de/trueffel-wissen/arten/wintertrueffel/	6	79	0.07594937	10.822785
9	https://dein-trueffel.de/trueffel-wissen/trueffelsuche/	48	576	0.08333333	8.373264
10	https://dein-trueffel.de/trueffel-wissen/trueffelsuche/trueffelhu...	4	125	0.03200000	13.056000
11	https://dein-trueffel.de/trueffel-wissen/trueffelsuche/trueffelsc...	4	273	0.01465201	7.736264
12	https://dein-trueffel.de/trueffel-wissen/wo-wachsen-trueffel/	212	1178	0.17996604	6.238540

Abb.6: Top-10-URLs nach Clicks

nur eine Impression generiert, sollte sie in Relation zu Ihren anderen Seiten mit mehr Impressions und ggf. schlechterem Ranking nicht als Ihre Top-Seite qualifiziert werden. Daher gewichten Sie für die Spalte *weight_position* die durchschnittliche Position anhand der Impressions. Als Resultat erhalten Sie eine reduzierte Tabelle, in der je URL nur noch eine Zeile vorhanden ist. Die Datumsspalte entfällt automatisch.

Nach dem *summarise()* folgt noch ein *ungroup()*, welches die internen Gruppen wieder auflöst.

Der DataFrame sieht nun wie in Abbildung 5 aus. Für die Beispiel-Website sind insgesamt 15 URLs übrig geblieben, die in der vergangenen Woche Impressions generiert haben. Sie wollen in Ihrem Report aber nur die Top-10-URLs nach Clicks anzeigen. Zum Glück gibt es dafür eine *top_n()*-Funktion (3). Das Argument n nimmt die Anzahl der zu behaltenden Zeilen an. Mit dem Argument *wt* definieren Sie, anhand welcher Spalte die Auswahl der Zeilen erfolgen soll. Standardmäßig werden somit die zehn Zeilen mit den höchsten *sum_clicks*-Werten ausgewählt. Wollen Sie die URLs mit den schlechtesten Click-Werten, können Sie ein Minus vor die Spalte schreiben (*wt = -sum_clicks*). Weil einige URLs die gleiche Anzahl an Clicks erhalten haben, finden sich in der Tabelle allerdings zwölf statt zehn Zeilen.

Ihre Tabelle sieht zum jetzigen Zeitpunkt aus, wie in Abbildung 6 zu sehen. Da Sie die Tabelle in Ihrem Report gerne absteigend nach Clicks sortieren wollen, führen Sie im Anschluss die Funktion *arrange()* aus (4). In dieser geben Sie die zu sortierende Spalte an. Das *desc()* (descending) zeigt die absteigende Sortierung an. Lassen Sie es weg, erfolgt die Sortierung automatisch absteigend.

Nachdem Sie den DataFrame in Form gebracht haben, widmen Sie sich der Bearbeitung der Werte (5). Mit der Funktion *mutate()* können Sie bestehende Zeilen bearbeiten oder neue hinzufügen. Im vorliegenden Fall wollen Sie die Spalten *page, ctr* und *weight_position* bearbeiten. Zuerst schneiden Sie die Domain aus den URLs ab. In der Tabelle des finalen Reports nimmt sie nur Platz weg. Sie bedienen sich dazu der Funktion *str_replace()*. Als erstes Argument nimmt sie die Spalte auf, in der ein String ersetzt werden soll. Das zweite Argument pattern ist das Muster, das ersetzt werden soll. Hier geben Sie die Variable *GSC_PROP* an, die Sie bereits im ersten Artikel dazu verwendet haben, die GSC-Daten von der API abzufragen; *replacement* nimmt dann das einzusetzende Zeichen an. Konkret ersetzen Sie dadurch bei allen URLs „*https://dein-trueffel.de/*“ durch „/“. Danach runden Sie die Werte der Spalten *ctr* und *weight_position* auf drei bzw. zwei Nachkommastellen.

Zum Schluss bringen Sie noch etwas Hochglanz auf. Die Spaltennamen sind aktuell noch normalisiert, sprich, kleingeschrieben und mit Unterstrichen. Da das in Reports nicht unbedingt schön aussieht, benennen Sie sie mittels rename() um (6). Ihre finale Tabelle sehen Sie in Abbildung 7.

Top-Queries aus der Google Search Console ermitteln

Geschafft! Nachdem Sie diese Tour de Force überstanden haben, lernen Sie einen der wesentlichen Vorzüge der Programmierung mit R kennen. Sie können Code-Abschnitte einfach wiederverwenden. Denn für die Berechnung der Top-10-Queries ist der Ablauf identisch (Abbildung 8). Sie müssen nur den anfänglichen DataFrame zu *gsa_dim_date_query* ändern. Dann noch eine kleine Anpassung im *group_by()*, denn jetzt aggregieren Sie ja auf Query- und nicht mehr auf

URL-Basis. Zu guter Letzt ändern Sie die Benennung des DataFrames, in den die Tabelle geschrieben werden soll, zu *gsa_top_10_queries*. Die Tabelle mit den Top-Queries sehen Sie in Abbildung 9.

S-DataFrames zusammenführen

Nun können Sie sich dem Zusammenführen der DataFrames mit den Sichtbarkeitswerten (SI) widmen. Fügen Sie dazu den Code aus Abbildung 10 in Ihr Skript ein. Der DataFrame *si_desktop*, der die Sichtbarkeitsdaten der Desktop-Variante enthält, sieht aktuell aus, wie in Abbildung 11 zu sehen. Bevor Sie diesen DataFrame mit *si_mobile* zusammenführen, gilt es noch drei Transformationen vorzunehmen (Abbildung 10; 1): Zunächst fällt auf, dass die date-Spalte das Datum mit Zeitstempel enthält. Diese sind für den späteren Report überflüssig und werden daher zu einem reinen Datumswert reduziert. Dazu greifen Sie erneut auf die Funktion *mutate()* zurück. In dieser wenden Sie die Funktion *date()*, die das Datum extrahiert, auf die Spalte date an. Den so gewonnenen Wert schreiben Sie in dieselbe Spalte zurück. Sie überschreiben somit die bestehenden Werte einfach. Anschließend entfernen Sie mittels *select()* die Spalte domain. Das vorangestellte Minus (-domain) zeigt eine negative Auswahl an, sprich, die Funktion soll alle Spalten außer domain auswählen. Schließlich benennen Sie durch *rename()* die Spalte value zu Desktop um. Dadurch machen Sie explizit, dass es sich bei den Werten dieser Spalte um die Desktop-SI handelt. Würden Sie die Spalte nicht umbenennen und im nächsten Schritt die Mobile-Werte an die Tabelle schreiben, hätten Sie zwei Spalten value – und könnten folglich nicht ohne Weiteres sagen, welche Spalte die Desktop- resp. die Mobile-Werte enthält.

Und damit kommen Sie auch schon zur Funktion, die die beiden DataFrames miteinander verbindet (2). Konkret wollen Sie die Spalte mit den mobilen SI aus dem DataFrame *si_mobile* als zusätzliche Spalte an den DataFrame *si_desktop* schreiben, sodass eine Zeile je Datumswert den Desktop- und den Mobile-SI listet. Vergleichbar ist diese Operation, die man als Join bezeichnet, mit einem *SVERWEIS* in Excel. Sie ist allerdings etwas komplexer, denn die beiden DataFrames weisen unterschiedliche Datumsbereiche auf. Der Desktop-SI wird seit 2008, der Mobile-SI erst seit 2015 reportet. Für jeden Datumspunkt des Desktop-SI gibt es somit nicht unbedingt eine entsprechende Zeile im Mobile-SI. Da Sie im Rahmen von Analysen sehr häufig

	Page	Clicks	Impressions	CTR	Position
1	/trueffel-wissen/wo-wachsen-trueffel/	212	1178	0.180	6.24
2	/trueffel-produkte/trueffelbutter/	107	1033	0.104	5.43
3	/trueffel-wissen/trueffelsuche/	48	576	0.083	8.37
4	/trueffel-rezepte/383-4/	17	143	0.119	8.58
5	/trueffel-wissen/arten/china-trueffel/	14	87	0.161	9.64
6	/trueffel-wissen/arten/perigord-trueffel/	10	243	0.041	9.23
7	/trueffel-wissen/arten/wintertrueffel/	6	79	0.076	10.82
8	/	5	15	0.333	5.73
9	/trueffel-produkte/trueffelhonig/	4	6	0.667	5.67
10	/trueffel-rezepte/383-3/	4	44	0.091	36.91
11	/trueffel-wissen/trueffelsuche/trueffelhund/	4	125	0.032	13.06
12	/trueffel-wissen/trueffelsuche/trueffelschwein/	4	273	0.015	7.74

Abb.7: Die finale Tabelle gsa_top_10_pages

```
230
231 # Query: Berechnung der Top 10 Queries nach Clicks der Vorwoche
232 gsa_top_10_queries <- gsa_dim_date_query %>%
233   filter(between(date, MONDAY_PREVIOUS_WEEK, SUNDAY_PREVIOUS_WEEK)) %>%
234   group_by(query) %>%
235   summarise(sum_clicks = sum(clicks),
236             sum_impressions = sum(impressions),
237             ctr  = sum_clicks / sum_impressions,
238             weight_position = sum(impressions * position) / sum(impressions)) %>%
239   ungroup() %>%
240   top_n(10, sum_clicks) %>%
241   arrange(desc(sum_clicks)) %>%
242   mutate(ctr = round(ctr, 3),
243          weight_position = round(weight_position, 2)) %>%
244   rename("Query" = query,
245          "Clicks" = sum_clicks,
246          "Impressions" = sum_impressions,
247          "CTR" = ctr,
248          "Position" = weight_position)
```

Abb.8: Berechnung der Top-10-Queries nach Clicks der Vorwoche

	Query	Clicks	Impressions	CTR	Position
1	trüffelbutter	62	563	0.110	2.75
2	wo wachsen trüffel	35	78	0.449	1.22
3	wo findet man trüffel	14	30	0.467	1.90
4	trüffelsuche	13	48	0.271	1.48
5	trüffel finden	8	21	0.381	2.38
6	wo wächst trüffel	8	15	0.533	1.40
7	trüffelbutter verwendung	6	10	0.600	1.20
8	trüffel in deutschland	5	24	0.208	7.50
9	trüffel fundorte	4	8	0.500	1.12
10	trüffelbutter rezept	4	19	0.211	7.63

Abb.9: Die finale Tabelle gsa_top_10_queries

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

si ← si_desktop %>%
  mutate(date = date(date)) %>%
  select(-domain) %>%
  rename("Desktop" = value) %>%

  left_join(si_mobile %>%
              mutate(date = date(date)) %>%
              select(-domain) %>%
              rename("Mobile" = value),
            by = "date") %>%

  replace_na(list(Mobile = 0)) %>%

  filter(between(date, START_DATE, END_DATE)) %>%

  gather(key = "platform",
         value = "value",
         -date)
```

Abb.10: Zusammenführen der Sichtbarkeits-DataFrames

Tabellen miteinander verbinden werden, finden Sie in Abbildung 12 eine Übersicht der gängigsten Joins.

Im vorliegenden Beispiel wissen Sie, dass der DataFrame *si_desktop* mehr Zeilen enthält als *si_mobile*. Sie wenden daher einen *left_join()* an, denn Datumspunkte in *si_desktop*, die keine Entsprechung *si_mobile* aufweisen, sollen nicht entfallen (dies wäre bei einem *inner_join()* der Fall), sondern zunächst mit *NAs* aufgefüllt werden und damit erhalten bleiben. Der *left_join()* ist nun wie folgt aufgebaut: Zunächst nimmt die Funktion als erstes Argument den DataFrame auf, der mit der si_desktop zusammengeführt werden soll. Dies ist der *si_mobile*. Verschachtelt, innerhalb des *inner_join()*, transformieren Sie diese Tabelle analog zu *si_desktop*, mit dem Unterschied, dass Sie die Spalte value nun zu Mobile umbenennen. Das Argument by gibt an, welche Spalte als Schlüssel (Key) dienen soll, anhand dessen die beiden DataFrames miteinander gejoint werden. Hier wählen Sie die Spalte date, denn Sie wollen,

HINWEIS

Haben Sie Fragen? Dann stellen Sie sie gerne in der Facebook-Gruppe *https://www.facebook.com/groups/ompyr/*. So können alle von den Antworten profitieren. Seien Sie versichert, es gibt keine „dummen" Fragen, denn aller Anfang ist schwer.

dass für jedes Datum der Desktop- und der Mobile-SI in einer Zeile stehen. Die zunächst mit *NA* aufgefüllten Werte in der Mobile-Spalte ersetzen Sie anschließend durch die bereits bekannte Funktion *replace_na()* mit 0 (Abbildung 10; 3).

Für Ihren Report haben Sie definiert, dass Sie nur einen Berichtszeitraum von rückwirkend sechs Monaten verwenden möchten (*START_DATE* und *END_DATE* aus Teil 1 stehen am Anfang des Skripts). Über die Filterfunktion schränken Sie den DataFrame auf diesen Bereich ein (4). Damit machen Sie im vorliegenden Beispiel den vorherigen Schritt des Auffüllens der mobilen SI-Werte natürlich im Grunde überflüssig, da Sie die Zeilen direkt wieder verwerfen. Dass Sie diesen Schritt jedoch durchgeführt haben, bietet den Vorteil, dass Sie fortan den Berichtszeitraum nach Belieben in die Vergangenheit vergrößern können.

Zum Schluss überführen Sie den nun weiten DataFrame mittels *gather()* in einen langen (5), damit er eine zum Plotting geeignete Struktur hat. Der finale DataFrame *si* sieht nun aus, wie in Abbildung 13 zu sehen.

si_desktop

Filter

	domain	date	value
1	dein-trueffel.de	2019-01-28T00:00:00+01:00	0.0002
2	dein-trueffel.de	2019-01-21T00:00:00+01:00	0.0002
3	dein-trueffel.de	2019-01-14T00:00:00+01:00	0.0015
4	dein-trueffel.de	2019-01-07T00:00:00+01:00	0.0002
5	dein-trueffel.de	2018-12-31T00:00:00+01:00	0.0002
6	dein-trueffel.de	2018-12-24T00:00:00+01:00	0.0002
7	dein-trueffel.de	2018-12-17T00:00:00+01:00	0.0000

Abb.11: DataFrame si_desktop

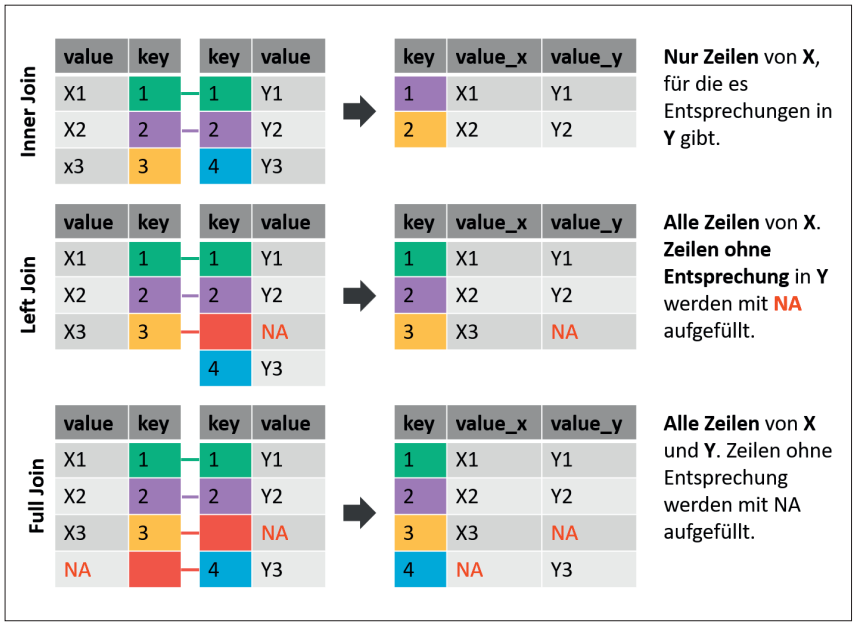

Abb.12: Übersicht von Join-Operationen

Fazit

Jetzt bleibt nur noch eines übrig: Klopfen Sie sich kräftig auf die Schulter und vergegenwärtigen Sie sich noch einmal, was Sie alles geschafft und gelernt haben!

Sie haben die Top-10-Seiten nach GSC-Clicks berechnet. Dabei haben Sie die zwei zentralen Funktionen zur Datenaggregation kennengelernt: *group_by()* und *summarise()*. Kein Report und keine Analyse kommen ohne diese beiden aus, denn sie ermöglichen es Ihnen, einen DataFrame entsprechend den Ausprägungen in einer Spalte zu gruppieren und die einzelnen Gruppen gesondert voneinander zu aggregieren. Ein simples Beispiel aus einem anderen Bereich ist das Zählen der einzelnen Status-Codes in einem Crawl oder die Berechnung der durchschnittlichen Verweildauer auf den Seiten der einzelnen Kategorien in einem Blog.

Sie haben einen kurzen Abstecher in die String-Manipulation absolviert, indem Sie mittels *str_replace()* die Domain aus den URLs entfernt haben. Replace ist eine mächtige, aber bei Weitem nicht die einzige Funktion in diesem Bereich. Geben Sie einmal *str_* auf der Console ein und scrollen Sie durch die Autovervollständigung. Sie finden eine Fülle von Funktionen, um Bestandteile von Strings zu entfernen, zu transformieren oder zu extrahieren. Gerade beim Arbeiten mit URLs ist das ein unverzichtbares Werkzeug, um bspw. irrelevante Parameter abzuschneiden, URLs der mobilen auf die Desktop-Variante umzuschreiben oder Produkt-IDs zu extrahieren.

Schließlich haben Sie einen Join zweier DataFrames durchgeführt – ebenfalls ein elementares Werkzeug bei der Arbeit mit Daten, denn fast immer müssen diese aus verschiedenen Quellen zusammengeführt werden. ¶