Foto: iunewind / gettyimages.de

Patrick Lürwer

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

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.

In den vorherigen Teilen dieser Serie haben Sie Daten von der Google-Analytics- und Search-Console- sowie Sistrix-API heruntergeladen. Diese Daten haben Sie anschließend transformiert, um fehlende Datumspunkte zu ergänzen, zu aggregieren und zusammenzuführen. Damit haben Sie die Basis für die danach folgende Visualisierung gelegt, der Sie sich in diesem Teil widmen werden. Sie werden dabei das Konzept der Grammar of Graphics kennenlernen und wie Sie mit ggplot2 verschiedene Diagrammtypen erstellen können.

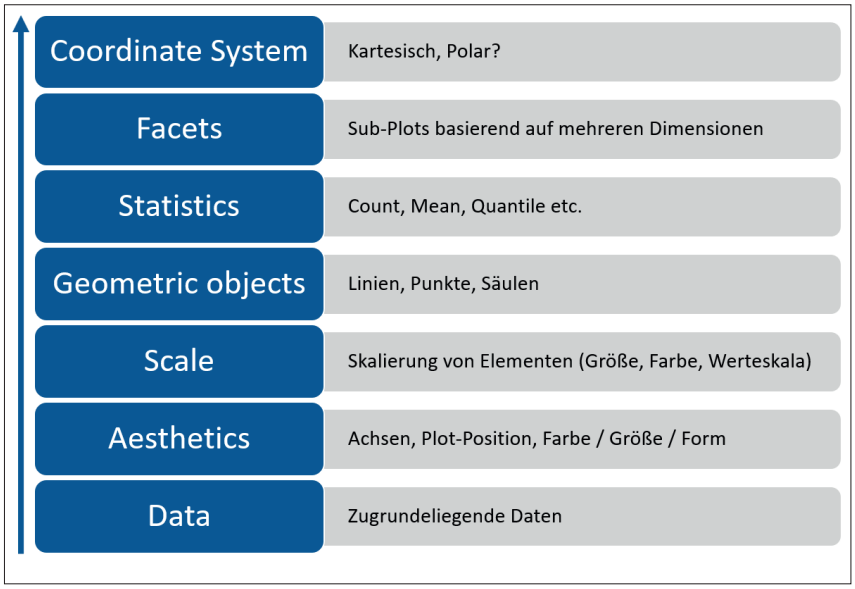

Abb.1: Schichten eines Plots nach der Grammar of Graphics

Bisher haben Sie „nur" mit Tabellen, sprich DataFrames, gearbeitet. Die Plots, die Sie in den bisherigen Teilen gesehen haben, dienten bislang allein der Veranschaulichung von Datentransformationen. Und damit ist auch bereits der zentrale Zweck dieses Artikels genannt, denn so abgedroschen die Phrase klingen mag, behält sie doch ihre Gültigkeit: „Ein Bild sagt mehr als tausend Worte." Menschen können visualisierte Daten intuitiv besser erfassen als die zugrunde liegenden tabellarischen Daten. Sie können sich zwar durchaus die Sichtbarkeit Ihrer Website im zeitlichen Verlauf Zeile für Zeile in einer Tabelle durchlesen. Um schnell einen Trend identifizieren zu können, ist ein klassisches Liniendiagramm, welches das Datum auf der x- und die Sichtbarkeit auf der y-Achse aufträgt, allerdings jeder tabellarischen Darstellung weit überlegen. Entsprechend werden Sie auch vorwiegend Diagramme in Ihrem Report verwenden, die Sie im Nachfolgenden mit ggplot2 erstellen werden.

ggplot2 und Grammar of Graphics

Ggplot2 *(https://ggplot2.tidyverse.org/)* ist Bestandteil des tidyverse und basiert auf der Grammar of Graphics von Leland Wilkinson, die von Hadley Wickham für sein R-Package adaptiert wurde *(http://vita.had.co.nz/papers/layered-grammar.html)*. Jene ist ein Framework, welches einen schichtenbasierten Ansatz verfolgt, um Visualisierungen in einer strukturierten Art zu beschreiben und zu konstruieren. Damit dieses abstrakte Konzept verständlicher wird, sehen Sie in Abbildung 1, welche Bestandteile die einzelnen Schichten einer Grafik bilden.

1. **Data**: Die zugrunde liegenden Daten, die visualisiert werden sollen.
2. **Aesthetics**: Welche Spalten eines DataFrames sollen auf welcher Achse abgetragen werden? Sollen zusätzliche Dimensionen der einzelnen Datenpunkte durch Farbe, Größe oder Form codiert werden?
3. **Scale**: Sollen bestimmte Elemente skaliert werden? Bspw. logarithmische Skalierung der Achsen, bestimmte Farbcodierung oder Größe der Datenpunkte.
4. **Geometric objects**: Wie sollen die Daten dargestellt werden? Sollen sie als Linien-, Punkt- oder Balkendiagramm visualisiert werden?
5. **Statistics**: Sollen die Daten statistisch verarbeitet werden? Bspw., indem die Anzahl gezählt oder der Durchschnitt gebildet wird.
6. **Facets**: Sollen Sub-Plots basierend auf bestimmten Dimensionen der Daten gebildet werden?
7. **Coordinate System**: Soll das Koordinatensystem des Plots kartesisch (rechteckiges Gitter mit x- und y-Achse) oder polar (kreisförmiges Gitter) dargestellt werden?

Konstruktion eines Plots

Damit Sie sehen, wie die tatsächliche Komposition dieser Schichten aussieht, können Sie den Code in Abbildung 2 in einem separaten Skript ausführen. Sie visualisieren damit exemplarisch den Datensatz *mtcars*, der standardmäßig in R enthalten ist und neben anderen Kennzahlen den Treibstoffverbrauch von 32 Autos beinhaltet. Mit *data()* laden Sie den Datensatz in Ihr Environment.

Zunächst ziehen Sie mit *ggplot()* eine Leinwand auf, auf der das Diagramm dargestellt werden soll (1). Führen Sie die Funktion aus, sehen Sie in Ihrem Plot-Viewer zunächst nur eine graue Fläche, denn es fehlen jegliche Angaben zu den darzustellenden Daten. Diese fügen Sie nun Schritt für Schritt hinzu.

Bei (2) geben Sie den zu verwendenden Datensatz an. Auch hier sehen Sie weiterhin nur eine graue Fläche, denn nun fehlen noch die Angaben, welche Spalten des DataFrames auf welcher Achse dargestellt werden sollen.

Dies geben Sie bei (3) mit *aes()* an, indem Sie definieren, dass das Gewicht (wt; weight in 1000 lbs) auf der x- und der Kraftstoffverbrauch (*mpg*; Miles per gallon) auf der y-Achse aufgetragen werden sollen. In Abbildung 3 sehen Sie das Ergebnis. Noch werden keine Datenpunkte angezeigt, denn Sie haben

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.

noch nicht definiert, in welcher Form dies geschehen soll. Sie sehen aber bereits, dass das Achsen-Spektrum entsprechend den Werten im DataFrame aufgezogen wurde.

In (4) geben Sie nun die Schicht Geometric object, also die Darstellungsform, an. Konkret möchten Sie damit bezwecken, dass die einzelnen Daten als Punkte geplottet werden (Abbildung 4).

Nun folgt die optionale Skalierung der Elemente (5). Innerhalb der Aesthetics (*aes()*) geben Sie an, dass die Größe der Punkte entsprechend dem Wert *hp* (horse power) skaliert werden soll. Die konkrete Skalierung regeln Sie über *scale_size()*, indem Sie die Größe der Punkte innerhalb eines Spektrums von 1 bis 10 Einheiten zulassen (Abbildung 5).

Analog dazu können Sie auch die Farbe der Punkte anhand einer weiteren Spalte bestimmen (6). Im Beispiel sollen die Punkte entsprechend den Gängen (*gear*) eingefärbt werden. Dazu fügen Sie in *aes()* noch das Argument color ein und mappen darauf die Spalte gear. Das umfassende *factor()* zeigt ggplot2 an, dass es sich hier um kategoriale Daten handelt. Ohne das *factor()* würden die Farben in einem kontinuierlichen Farbspektrum ausgewählt, da die zugrunde liegenden Daten numerisch sind. Löschen Sie ruhig das *factor()* und schauen Sie sich den Unterschied an. Mittels *scale_color_manuell()* bestimmen Sie die zu verwendenden Farbwerte selbst (Abbildung 6). Grundsätzlich können Sie die beiden *scale_**-Funktionen auch weglassen. Dann verwendet ggplot2 Default-Werte für die Größe resp. die Farben.

Im Anschluss (7) facettieren Sie die Grafik nach einer weiteren Spalte, in diesem Fall anhand der Anzahl der Gänge (*gear*) (Abbildung 7). Anders ausgedrückt, unterteilen Sie den initialen Datensatz in drei Gruppen, die in eigenständigen Diagrammen dargestellt werden.

```
1
2  library(tidyverse)
3
4  data("mtcars")
5
1  # Leinwand
   ggplot()
8
2  # Dataset
   ggplot(data = mtcars)
11
3  # Aesthetics
   ggplot(mtcars, aes(x = wt, y = mpg))
14
15 # Geometric objects
4  ggplot(mtcars, aes(wt, mpg)) +
17   geom_point()
18
19 # Scales: Größe
5  ggplot(mtcars, aes(wt, mpg, size = hp)) +
21   geom_point() +
22   scale_size(range = c(1,10))
23
24 # Scales: Größe & Farbe
25 ggplot(mtcars, aes(wt, mpg, size = hp, color = factor(gear))) +
6    geom_point() +
27   scale_size(range = c(1,10)) +
28   scale_color_manual(values = c("#377EB8", "#FF7F00", "#007500"))
29
30 # Facets
31 ggplot(mtcars, aes(wt, mpg, size = hp, color = factor(gear))) +
33   geom_point() +
7    scale_size(range = c(1,10)) +
34   scale_color_manual(values = c("#377EB8", "#FF7F00", "#007500")) +
35   facet_wrap(~cyl)
36
37 # Statistics
38 ggplot(mtcars, aes(x = factor(cyl))) +
8    stat_count()
41
42 ggplot(mtcars, aes(hp)) +
43   stat_bin()
44
45 # Feintuning
46 ggplot(mtcars, aes(wt, mpg, size = hp, color = factor(gear))) +
47   geom_point() +
48   scale_size(range = c(1,10)) +
   facet_wrap(~cyl) +
9  theme_light() +
51   labs(title = "Kraftstoffverbrauch ~ Gewicht",
52        subtitle = "je Ganganzahl",
53        x = "Gewicht",
54        y = "Miles / Gallon",
55        size = "PS",
56        color = "Gänge")
```

Abb. 2: Konsekutiver Aufbau eines Beispiel-Plots

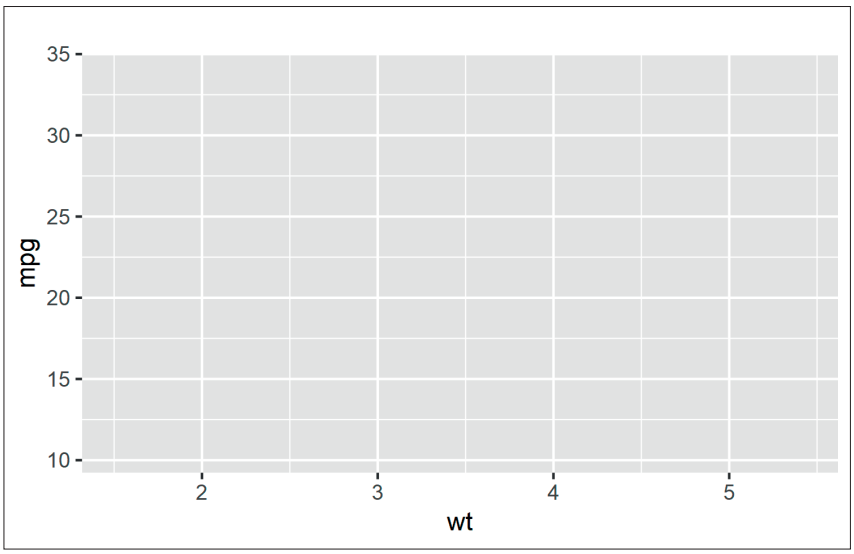

Abb.3: Leinwand mit definierten Achsen

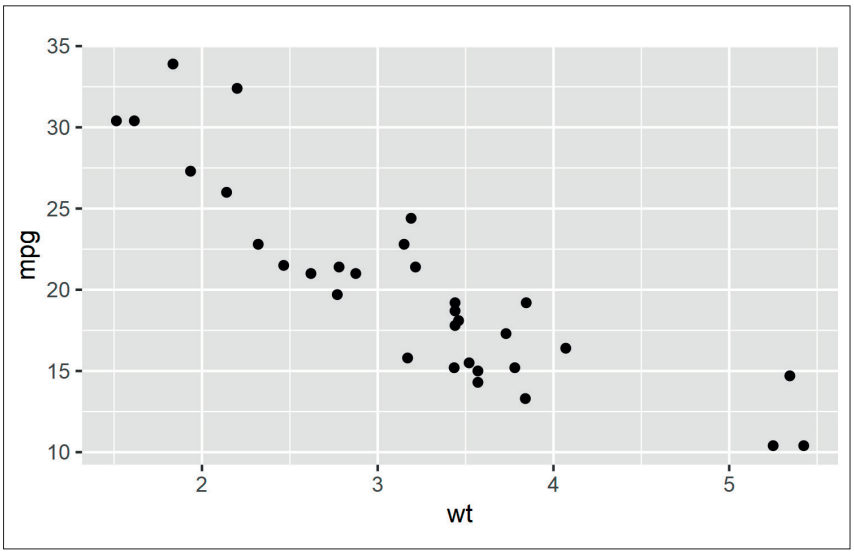

Abb.4: Visualisierung der Daten als Punkte

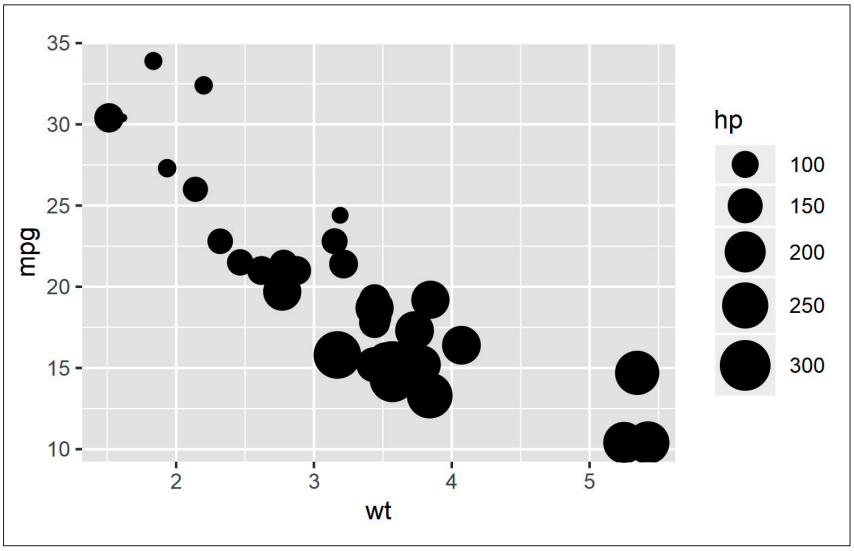

Abb.5: Skalierung der Punktgröße entsprechend den Pferdestärken (hp)

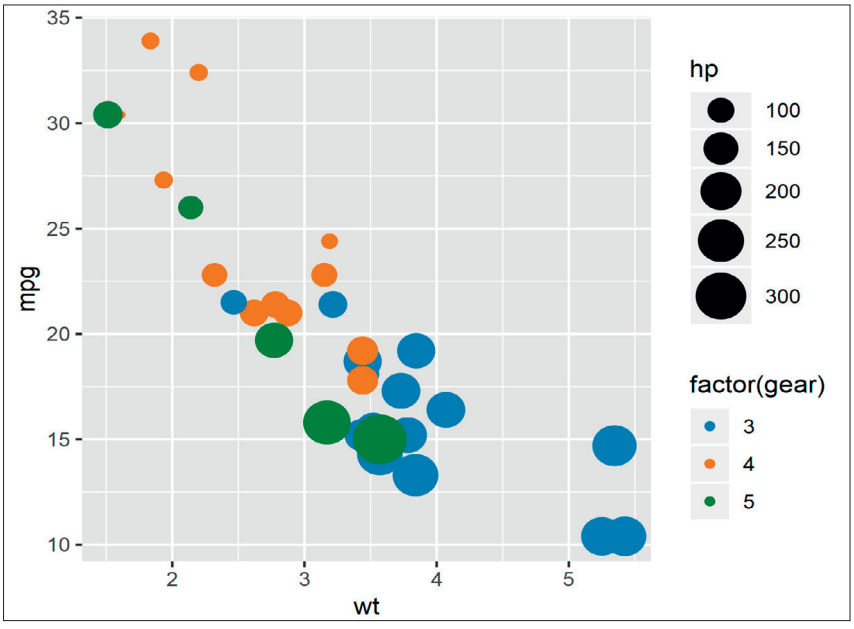

Abb. 6: Skalierung der Farbwerte entsprechend der Zylinderanzahl (cyl)

Zur Veranschaulichung der Statistics-Schicht nehmen Sie vorübergehend eine andere Spalte für die x-Achse (8), da die bisherigen x- und y-Werte keine sinnvolle statistische Berechnung zulassen. *stat_count()* dient dazu, die Anzahl der jeweiligen Ausprägungen in der Spalte *cyl* zu zählen, sprich: Wie viele Autos haben vier, sechs bzw. acht Zylinder. Dieses Zählen erfolgt automatisch im Hintergrund, ohne dass Sie die Aggregationstabelle tatsächlich sehen können. Als Ergebnis erhalten Sie ein klassisches Säulendiagramm (Abbildung 8). Löschen Sie auch hier ruhig einmal das umgebende *factor()* aus dem Code und sehen Sie sich den Unterschied an. Eine weitere, häufig genutzte Statistic-Funktion ist *stat_bin()*, die ein Histogramm plottet. Dadurch können Sie sich die Verteilung der Pferdestärken (*hp*) ansehen, indem die PS in verschiedenen Töpfen (*bins*) zusammengefasst und die darin enthaltenen Elemente gezählt werden (Abbildung 9).

Zurück zum bisherigen Plot. Natürlich können Sie nicht nur Daten mappen, sondern auch das Aussehen des Plots an sich verändern. In (9) ändern Sie zunächst das Theme. In diesem Fall wählen Sie mit *theme_light()* ein helleres aus. Probieren Sie verschiedene Themes aus, indem Sie theme_ eingeben und dann ein konkretes Theme aus der Autovervollständigung auswählen. Via *labs()* definieren Sie die Beschriftung des Plots. Die ersten Argumente sind selbsterklärend. Durch *size* = geben Sie an, wie die Überschrift der Legende für die *size*-Aesthetic benannt werden soll. Analog dazu erfolgt die Umbenennung der *color*-Legende. Das Ergebnis sehen Sie in Abbildung 10.

Mit dieser „kleinen" Einführung an der Hand sind Sie für die nachfolgende Visualisierung Ihrer Report-Daten gewappnet. Denn das zugrunde liegende Konzept der Schichten ist immer gleich, was der große Vorteil von *ggplot2* und der Grammar of Graphics ist. Zugege-

ben, die Einarbeitung ist mit Sicherheit zunächst etwas mühsam, denn man ist bspw. von Excel gewohnt, dass einem das Programm sehr viele Aspekte der Visualisierung automatisch abnimmt. Hat man das Konzept aber erst einmal verinnerlicht, bieten sich unbegrenzte Möglichkeiten. Mit *ggplot2* können sie nämlich jedes (!) Detail eines Plots selbst bestimmen. Das heißt aber auch, dass Sie selbst bei einfachen Plots fast jede Schicht selbst konfigurieren müssen. In dieser Hinsicht ist Excel wiederum deutlich schneller. Aber mit der Zeit werden Sie sich ein Repertoire an Plot-Templates angelegt haben, die Sie einfach nur noch mittels Copy & Paste einfügen.

Nun aber endlich zur Visualisierung Ihrer Report-Daten.

Plotting der Google-Analytics-Daten

Nun aber zurück zu Ihren Report-Daten. Erstellen Sie wieder ein neues Skript, fügen Sie den bisherigen Code der Artikelserie ein und speichern Sie das Skript als *visualization.R*.

Zunächst definieren Sie zwei Variablen, die Farbwerte aufnehmen und die Sie in Ihren Plots verwenden möchten (Abbildung 11). Indem Sie die Farbwerte an einer zentralen Stelle definieren, können Sie diese jederzeit bequem ändern, ohne jeden Plot-Code einzeln anzufassen.

Nun folgt der Code zum Plotting der Google-Analytics-Daten (GA).

Sie erinnern sich: Diese liegen im DataFrame *ga_sessions* und umfassen den gleitenden Mittelwert der Sessions für die Channels *Direct* und *Organic Search*. Schauen Sie sich die Tabelle an, um sich die Struktur noch einmal zu vergegenwärtigen. Dies erleichtert Ihnen zu verstehen, wie die Daten im Plot gemappt werden. Sie möchten GA-Daten, getrennt nach *Direct* und *Organic Search*, im zeitlichen Verlauf als Liniendiagramm plotten. Den Code dazu sehen Sie in Abbildung 12. Die Abstände

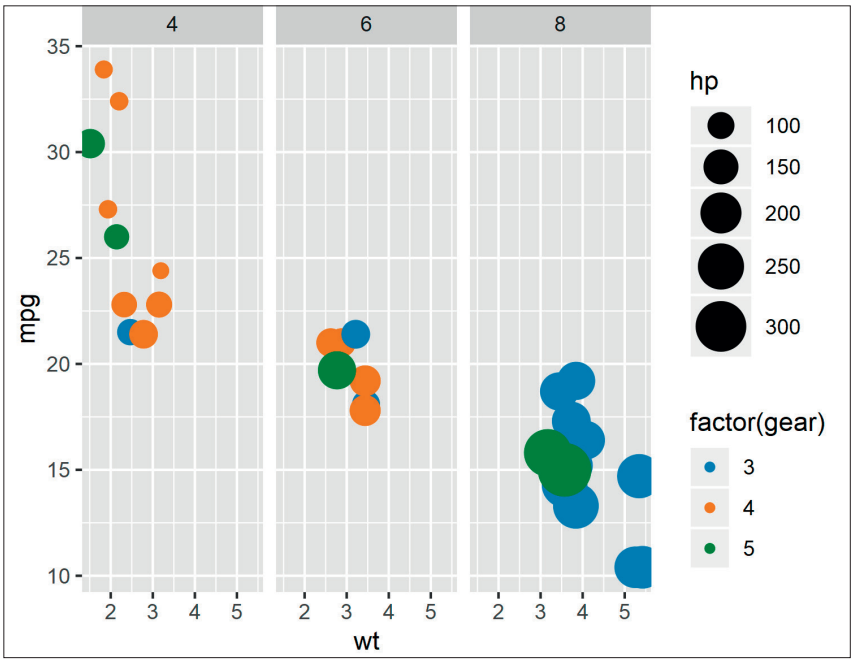

Abb. 7: Facettieren des Plots anhand einer zusätzlichen Daten-Dimension

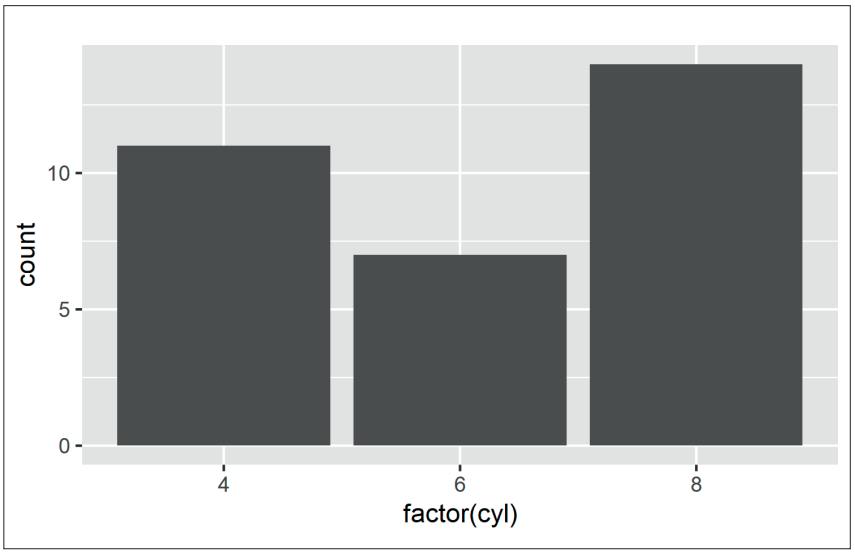

Abb.8: Säulendiagramm der Anzahl der jeweiligen Zylinder

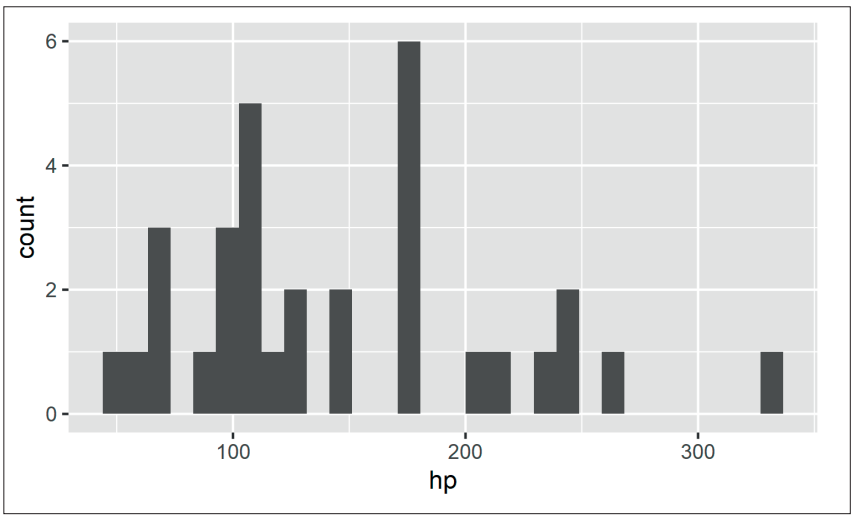

Abb. 9: Histogramm der Pferdestärken

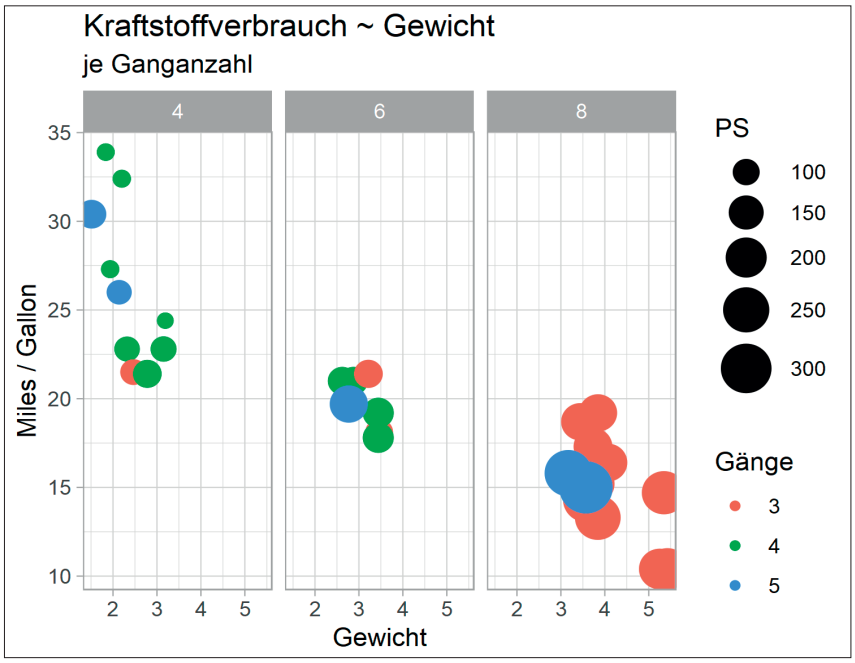

Abb. 10: Beschriftung des Plots

```
# GA ----------------------------------------------------------------------

ggplot(data = ga_sessions,
       aes(x = date,
           y = rollmean_sessions,
           color = channelGrouping)) +

  geom_line(size = 1) +

  geom_smooth(method = lm,
              se = TRUE,
              size = 0.5) +

  theme_minimal() +

  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(color = "#666666")) +

  labs(title = "Google Analytics - Sessions",
       subtitle = "Gleitender Mittelwert der Sessions je Channel-Grouping",
       x = "Kalenderwoche",
       y = "Sessions",
       color = "Channel Grouping") +

  scale_color_manual(values = c(COLOR_A, COLOR_B)) +

  scale_x_date(date_breaks = "2 weeks",
               date_labels = "%W\n'%y")
```

Abb.12: Plotting der GA-Daten

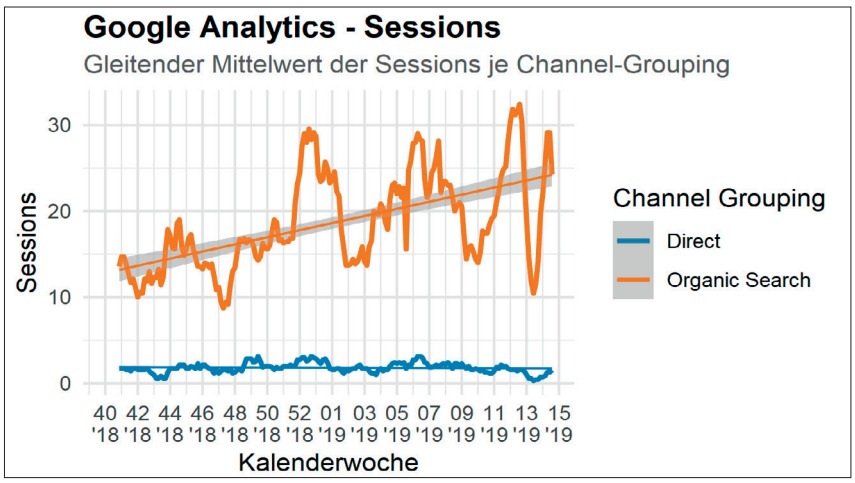

Abb.13: Plot der GA-Daten

```
# Visualisierung ----------------


# Konfiguration -----------------

COLOR_A <- "#377EB8"
COLOR_B <- "#FF7F00"
```

Abb.11: Farbwerte für Plots definieren

innerhalb des Codes dienen erneut der besseren Lesbarkeit und Abgrenzung der einzelnen Schichten. Führen Sie den Code ruhig Abschnitt für Abschnitt aus, um nachzuvollziehen, wie sich der Plot (Abbildung 13) zusammenbaut.

In (1) geben Sie die Daten sowie die Spalten an, auf denen basierend die x- und die y-Achse aufgezogen werden sollen. In diesem Fall sollen das Datum auf der x-Achse und die Sessions auf der y-Achse dargestellt werden. Bedenken Sie, dass Sie im DataFrame zwei Channel-Werte vorliegen haben. *Direct* oder *Organic Search* stehen als Ausprägungen in einer eigenen Spalte (*channelGrouping*). Daneben steht der zugehörige Tageswert der Sessions. Durch diese Struktur können Sie nun definieren, dass die Sessions-Linie entsprechend diesen beiden Dimensionsausprägungen aufgeteilt und eingefärbt werden soll (*color = channelGrouping*).

Die Visualisierung als Liniendiagramm geben Sie über *geom_line()* an. Die Linienstärke setzen Sie manuell auf 1, um sie von der nachfolgend hinzugefügten Trendlinie abzuheben (2).

Über die bestehenden Linien legen Sie jetzt mit *geom_smooth()* einfach eine weitere Linie (3). Hier sehen Sie einen großen Vorteil von ggplot2. Da alle Schichten aufeinander aufbauen, können Sie durchaus auch mehrere Elemente der Geometric-object-Schicht übereinanderlegen. Sowohl *geom_line()* als auch *geom_smooth()* greifen auf die Daten zurück, die Sie in *ggplot2()* angegeben haben. *geom_smooth()* zeichnet eine Trendlinie ein, die nach dem Linear-Regression-Modell gebildet wird (*method = lm*). *se = TRUE* zeigt an, dass sie ein Konfidenzintervall um diese Linie wünschen. Setzen Sie den Wert auf *FALSE* und sehen Sie sich den Unterschied an.

Standardmäßig verwendet *ggplot2* ein Theme mit grauem Hintergrund. Mit *theme_minimal()* geben Sie ein helleres vor (4).

In (5) nehmen Sie noch zwei Feineinstellungen am Theme vor. Innerhalb von *theme()* definieren Sie, dass der Plot-Title fett, der Untertitel in einem leichten Grau dargestellt werden soll. Die Beschriftung des Diagramms in (6) ist selbsterklärend.

Mittels *scale_color_manual()* weisen Sie den beiden Linien manuell die zuvor definierten Farbwerte *COLOR_A* und *COLOR_B* zu (7).

ggplot2 versteht sich von Haus aus sehr gut mit der Darstellung von Datumswerten auf einer Achse. Standardmäßig wird nicht das Datum an sich, sondern der ausgeschriebene Monat als Beschriftung der Achsen-Ticks geplottet. Dieses Verhalten können Sie noch feiner einstellen, indem Sie mittels *scale_x_date()* angeben, dass die Achsenabschnitte zweiwöchig (*date_breaks = „2 weeks“*) sein sollen (8). Als Datumsformat wählen Sie die Kalenderwoche (*%W*) und in der nächsten Zeile das verkürzte Jahr (*\n'%y*). Damit schaffen Sie Platz auf der y-Achse, sodass sich die Datumsangaben nicht gegenseitig überlagern.

Fertig! Den ersten Plot haben Sie erfolgreich gemeistert. Allerdings sehen Sie bei diesem ein Problem. Das Wertespektrum der beiden Dimensionsausprägungen unterscheidet sich deutlich. Da signifikant weniger Direct-Traffic auf die Seite kommt, wird die Linie durch den Organic-Search-Traffic sehr nah an die x-Achse gedrückt, sodass Veränderungen der Linie kaum noch sichtbar sind. Eine Lösung dieses Problems gehen wir bei den GSC-Daten an.

Plotting der Google-Search-Console-Daten

Für die Visualisierung der GSC-Daten können Sie den vorherigen Plot-Code weitestgehend wiederverwenden (Abbildung 14). Den Plot sehen Sie in Abbildung 15.

```
# GSC ----------------------------------------------------------------

ggplot(gsa_dim_date, aes(date, value, color = metric)) +

  geom_line(size = 1) +

  geom_smooth(method = loess,
              size = 0.5,
              se = FALSE) +

  facet_grid(metric~.,
             scales = "free_y",
             labeller = labeller(metric = c(rollmean_clicks = "Clicks",
                                            rollmean_impressions = "Impressions"))) +

  theme_minimal() +

  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(color = "#666666")) +

  labs(title = "GSC - Traffic",
       subtitle = "Gleitender Mittelwert der Impressions / Clicks",
       x = "",
       y = "") +

  scale_color_manual(values = c(COLOR_A, COLOR_B)) +

  scale_x_date(date_breaks = "1 month", date_labels = "%b\n'%y") +

  guides(color = F)
```

Abb.14: Plotting der GSC-Daten

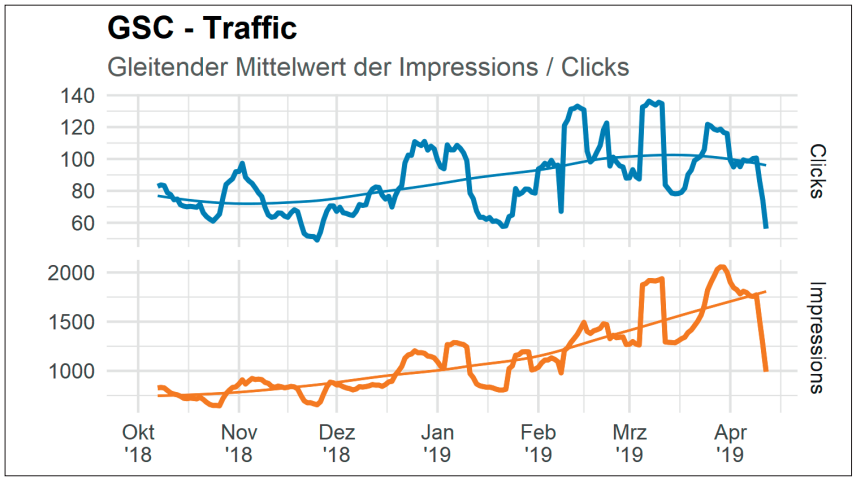

Abb.15: Facettierter Plot der GSC-Daten

Innerhalb von *ggplot()* geben sie den DataFrame *gsa_dim_date* als Datenbasis an (1). Die Linien sollen dieses Mal nach Clicks und Impressions aufgeteilt werden (*color = metric*). Hier sehen Sie nun auch, warum Sie im zweiten Teil der Artikelserie den DataFrame der GSC-Daten von einem weiten in ein langes Format überführt haben. Im weiten DataFrame standen die Clicks resp. die Impressions in eigenen Spalten. Um sie jedoch als Differenzierungsmerkmal für das Geometical object (*geom_line()*) verwenden zu können, müssen sie als Ausprägungen in einer Spalte stehen.

Zur Berechnung der Trendlinie verwenden Sie für diesen Plot die Methode Local Polynomial Regression Fitting (*method = loess*). Dadurch zeichnet die Trendlinie den Verlauf der zugrunde liegenden Daten näher nach, statt nur eine Gerade zu bilden, wie es beim Linear-Regression- Modell der Fall ist. loess trägt den Schwankungen in den Werten besser Rechnung, indem es den anfänglichen Rückgang der Clicks nachzeichnet. Wenn Sie loess durch lm ersetzen, sehen Sie bei den Clicks eine ausschließlich steigende Trendlinie. Das Konfidenzintervall unterdrücken Sie hier mittels *se = FALSE*.

Um der im GA-Plot sichtbaren Problematik entgegenzutreten, dass eine Linie undifferenzierbar nah an der x-Achse verläuft, weil ihr Wertespektrum deutlich unterhalb der anderen Linie liegt, facettieren Sie den Plot nach den beiden Metriken Clicks und *Impressions*,

```
# SI ----------------------------------------------------------------

ggplot(si,
       aes(date,
           value,
           color = platform,
           fill = platform)) +

  geom_area(alpha = 0.5) +

  geom_point() +

  facet_grid(platform~., scales = "free_y") +

  theme_minimal() +

  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(color = "#666666")) +

  labs(title = "Sistrix - Sichtbarkeitsindex",
       subtitle = "nach Platform",
       x = "",
       y = "SI") +

  scale_color_manual(values = c(COLOR_A, COLOR_B)) +

  scale_fill_manual(values = c(COLOR_A, COLOR_B)) +

  scale_x_date(date_breaks = "2 weeks", date_labels = "%W\n'%y") +

  guides(color = F,
         fill = F)
```

1 2 3 4 5 6 7

Abb.16: Plotting der SI-Daten

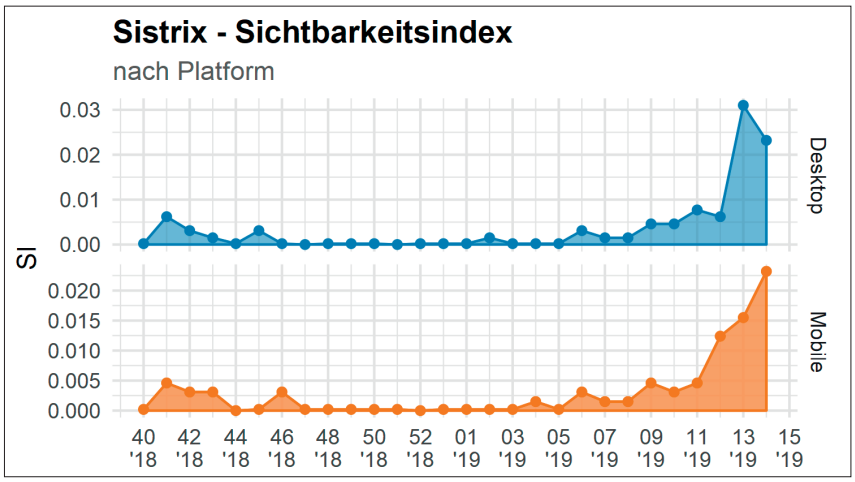

Abb.17: Plot der SI-Daten

um zwei unabhängige Diagramme zu erhalten (3). Innerhalb der Facetten skalieren Sie die y-Achsen separat *(scales = „free_y“)*, sodass sie sich nur bis zum Maximalwert der jeweiligen Metrik erstrecken. Aus der GSC kennen Sie mit Sicherheit die Darstellungsform, dass beide Linien in einem Diagramm mit unterschiedlich skalierter linker und rechter y-Achse gezeichnet werden. *ggplot2* lässt dies explizit nicht zu, da es in der Datenvisualisierung eine schlechte Praktik ist. (Die Begründung des Package-Autors Hadley Wickham können Sie hier nachlesen: *https://stackoverflow.com/a/3101876/4193607*.) Die Metriken, anhand derer Sie den Plot facettieren, haben im DataFrame die Benennungen *rollmean_clicks* und *rollmean_impressions*. Im DataFrame ist diese normalisierte und spezifische Benennung sinnvoll, um zweifelsfrei erkennen zu können, was die nebenstehenden Werte darstellen. Im Diagramm werden Sie dies über den Untertitel des Plots kommunizieren. Damit die Facetten-Überschriften einfacher zu lesen sind, benennen Sie sie manuell mittels *labeller()* um.

Als Variation skalieren Sie die x-Achse in Monatsabschnitten *(date_breaks = „1 month“)* und geben den Monatsnamen in gekürzter Form (*%b*) an (4).

Schließlich (5) geben Sie noch an, dass Sie keine Legende wünschen (*guides = F*). Da Sie die Linien mittels *color* eingefärbt haben, plottet *ggplot2* automatisch eine Legende entsprechend der Farbcodierung. Weil Sie dies aber aus rein optischen Zwecken machen – die beiden Metriken liegen ja nun in eigenen Facetten mit Überschrift –, ist eine Legende redundant.

Damit haben Sie auch die GSC-Daten visualisiert. Somit bleibt nur noch der Sistrix-Sichtbarkeitsindex (SI) zu plotten.

Plotting des Sistrix-Sichtbarkeitsindexes

Auch für diesen können Sie das bisherige Plot-Template wiederverwenden (Abbildung 16) – erneut mit einigen Variationen, um die Möglichkeiten von ggplot2 zu demonstrieren. Wie Sie in Abbildung 17 sehen, werden Sie dieses Mal ein sogenanntes Flächendiagramm erstellen, dass die Darstellung in der Sistrix-Toolbox nachbildet.

Wie gewohnt geben Sie in *ggplot()* die Datenbasis *si* an (1). Hinzu kommt, dass Sie nicht nur *color* verwenden, sondern auch *fill*. Letzteres dient dazu, die Fläche unterhalb der Linie einzufärben.

TIPP

Die „richtige“ Art der Datenvisualisierung ist ein spannendes Thema für sich. Wie können Diagramme gestaltet werden, damit sie eine Geschichte leicht verständlich erzählen? Wenn Sie dieses Thema interessiert und wenn Sie wissen möchten, warum Sie niemals Kuchendiagramme verwenden sollten, empfiehlt Ihnen der Autor wärmstens das Buch „Storytelling with Data“ von Cole Nussbaumer Knaflic und ihren Blog www.storytellingwithdata.com. Möchten Sie tiefer in die Diagrammerstellung mittels ggplot2 einsteigen, führt kein Weg an „ggplot2: Elegant Graphics for Data Analysis“ von Hadley Wickham vorbei.

Um ein Flächendiagramm zu generieren, verwenden Sie *geom_area()*, statt wie bisher *geom_line()* (2). Mittels *alpha = 0.5* geben Sie an, dass die Fläche transparent sein soll.

geom_point() wird als zusätzliche Schicht auf das Diagramm gelegt, um die Wochen als Punkte auf der Linie einzuzeichnen (3). Ihre Farbcodierung wird über *color* in den Aesthetics beeinflusst. Daher die Kombination aus *color* und *fill*.

Sie facettieren den Plot erneut nach einer Dimension (4). In diesem Fall nach den SI-Werten für Desktop und Mobile.

Da Sie in *aes()* sowohl *color* als auch *fill* verwendet haben, müssen sie für diese separat die zu verwendenden Farbwerte mittels *scale_color_manual()* resp. *scale_fill_manual()* angeben (5, 6).

Schließlich unterdrücken Sie auch in diesem Plot die Legende (7). Auch hier müssen Sie sowohl für *color* als auch für *fill* separat diese Angabe machen, da jedes Differenzierungsmerkmal seine eigene Legende erhält.

Spielen Sie auch hier etwas mit den Funktionen und Argumenten herum, um sich zu vergegenwärtigen, wie diese die Komposition des Plots im Einzelnen beeinflussen. Lassen Sie bspw. einmal eines der *geom_** weg, um zu sehen, welches Element im Plot aus welchem Geom resultiert.

Fazit

Das war's! Sie haben den letzten Schritt gemeistert, bevor sie alle bisherigen Elemente in einem Report vereinen werden. In diesem Teil haben Sie die Grundlagen der Grammar of Graphics in Kombination mit *ggplot2* kennengelernt, ein mächtiges, formalisiertes Framework zur Visualisierung aller Arten von Daten. Im Detail haben Sie verschiedene Geometrical objects wie Linien-, Flächen- und Punktdiagramme zur Visualisierung Ihrer Daten angewendet. Mittels verschiedener Methoden wurden in den Diagrammen Regressionslinien zur Trenderkennung eingezeichnet. Nicht zuletzt haben Sie erfahren, wie Sie die Textelemente (Achsen-Beschriftungen, Titel und Facetten-Überschriften) konfigurieren können. ¶

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.

IMPRESSUM

Herausgeber & Chefredakteur (verantwortlich):
Mario Fischer
E-Mail: redaktion@websiteboosting.com

Autoren dieser Ausgabe:
Andre Alpar, Dr. Martin Bahr, Alexander Beck, Jennifer Bölitz, Stephan Czysch, Raymond Eiber, Johanna Hartung, Timo Heinrich, Erich Kachel, Daniel Kaliner, Claudius Konopka, Olaf Kopp, Patrick Lürwer, Bastian Sens, Kaspar Szymanski

Anzeigenleitung:
Markus Lutz
E-Mail: anzeigenleitung@websiteboosting.com

Art Direction, Layout/Produktion:
Kai Neugebauer

Lektorat:
Ursula Wenke, www.lektorat-wenke.de

Fotos & Illustrationen:
Website Boosting / GettyImages
Titel: Website Boosting

Druck:
Vogel Druck und Medienservice GmbH
Leibnizstr. 5, 97204 Höchberg

Vertrieb:
PressUp GmbH
Postfach 70 13 11
22013 Hamburg
E-Mail: websiteboosting@pressup.de

Abonnement:
Website Boosting Aboservice
PressUp GmbH
Postfach 70 13 11
22013 Hamburg
Tel. 040 / 38 6666 – 342
Fax: 040 / 38 6666 - 299
E-Mail: websiteboosting@pressup.de

Erscheinungsweise: 6 x jährlich
Bezugspreis: Einzelheft: 9,80€
Bezugspreis Inland jährlich 51,00€ inkl. Versand
Bezugspreis Ausland jährlich 63,00€ inkl. Versand

Studenten im Inland erhalten gegen Vorlage einer Immatrikulationsbescheinigung einen Preisvorteil von 20%.

Verlagsleitung:
Michael Müßig
Tel: +49 931 / 26 038 04,
verlag@websiteboosting.com

Anschrift des Verlages
Hotspot Verlag GmbH
Obere Landwehr 4a, 97082 Würzburg
Tel: + 49 931 / 26 038 04
Fax: +49 931 / 26 038 05
E-Mail: verlag@hotspotverlag.de
www.hotspotverlag.de

Geschäftsführung:
Kai Neugebauer

Die Inhaber- und Beteiligungsverhältnisse lauten wie folgt: Gesellschafter zu 100% ist die Webvalue Holding GmbH

ISSN: 2191-6241