Datenanalyse mit R, Teil 1

by heise Developer • April 19, 2013 • original (http://www.heise.de/developer/artikel/Datenanalyse-mit-R-Teil-1-1845278.html?artikelseite=4)

- Beschreibende Statistik: beschreibt die Verteilung einer Variable, zum Beispiel die durchschnittliche Temperatur und deren Varianz in einer bestimmten Region.
- Testen statistischer Hypothesen: Hier wird üblicherweise eine "Nullhypothese" formuliert, die an Hand der Daten widerlegt werden soll. Beispielsweise könnte eine Nullhypothese lauten, dass das Herzinfarktrisiko mit dem Cholesterinspiegel nicht ansteigt.
- Vergleich unterschiedlicher Versuchsgruppen: untersucht, ob unterschiedliche Versuchsgruppen auch unterschiedliche statistische Eigenschaften haben. Beantwortet zum Beispiel die Frage, ob Männer und Frauen unterschiedliches Einkaufsverhalten aufweisen.
- Faktorenanalyse: findet häufig in der Auswertung von Umfragen Anwendung. Findet aus der großen Menge von Attributen die wesentlichen unbekannten Einflussfaktoren. In der Psychologie werden beispielsweise aus Fragebögen grundlegende Charaktereigenschaften hergeleitet.
- Clustering: ordnet Objekte in unterschiedliche Gruppen (Cluster), sodass alle Objekte innerhalb einer Gruppe ähnlich sind.
- Assoziationsanalyse: versucht aus einer Menge von Ereignissen herauszufinden, welche Ereignisse häufig zusammen auftreten. Klassisches Anwendungsgebiet ist die sogenannte Warenkorbanalyse, die ermittelt, welche Produkte häufig gemeinsam gekauft werden.
- Zeitreihenanalyse: untersucht die Entwicklung einer Variablen über die Zeit, um entweder zukünftige Entwicklungen vorherzusagen (zum Beispiel der Wertentwicklung bestimmter Finanzprodukte) oder aber um unvorhergesehene Änderungen aufspüren zu können (zum Beispiel für Störungsmeldungen).
- Regression: ermittelt anhand einer Menge von Trainingsdaten den Zusammenhang zwischen bestimmten

Objekteigenschaften und einer Zielgröße. Beispielsweise wird durch das Betrachten demographischer Daten und des bisherigen Kaufverhaltens eines Kunden auf dessen zukünftige Kaufbereitschaft geschlossen.

Das Konvertieren einer gesamten Spalte kann auch in einer Schleife erfolgen, wofür *Data Frame SO* nötig ist:

```
for(i in 1:length(SO$DeliveryDate))
      { SO$DeliveryDate[i] <-as.Date(SO$DeliveryDate[i], format="%Y-%m-%d") }
```

Tipp: Am besten legt man sich ein weiteres R-Skript Workspace.R an. Dort kann man die Befehle einfach eingeben. Mit dem [caps]Run[/cas]-Befehl des Editor-Fensters (oben rechts) lassen sich selektierte Zeilen ausführen. Zum Ausprobieren spart das viel Zeit.

Mit

```
> str(SO$DeliveryDate)
 chr [1:10] "15340" "15341" "15341" "15341" "15341" "15341" ...
```

ist zu erkennen, dass dies nicht ganz dem erwarteten Ergebnis entspricht. Um das zu verstehen, muss man wissen, dass R im Prinzip eine objektorientierte Skriptsprache ist, die das sogenannte Duck-Typing unterstützt. Nicht die Klasse eines Objekts bestimmt den Typ, sondern allein die Existenz einer Methode (siehe beispielsweise Duck-Typing (http://de.wikipedia.org/wiki/Duck-Typing)). Zwar führt das zu hoher Flexibilität, aber wie bei allen dynamischen Typsystemen zu einer erschwerten Fehlersuche, da der Typ sich nur zur Laufzeit überprüfen lässt. Es kommt hinzu, dass – im Gegensatz zu anderen Sprachen mit Duck-Typing wie Python, Ruby oder Smalltalk – in R intrinsische Typkonvertierungen durchgeführt werden, die Sprache sich also eher wie Fortran verhält.

Im Beispiel legt *as.Date()* zunächst ein neues Objekt der Klasse *Date* an, nimmt den Character-Vektor und versucht aus ihm, gemäß dem angegebenen Format, die Attribute der Date Class zu setzen. Anschließend wird einem Element des existierenden Character-Vektors (*SO$DeliveryDate*) dieser Wert durch den Assignment Operator <-

zugewiesen. Da ein Vektor allerdings immer nur einen bestimmten Typ enthalten kann und der existierenden Vektor vom Typ *character* ist, wird die neue Instanz der Date Class durch *coerce()* in einen Character-String konvertiert und dem Vektor *SO$DeliveryDate* zugewiesen.

Die (vermeintliche) Lösung für das Problem mit der for-Schleife wäre folglich, einen neuen Vektor des Typs Date und der Länge 1 anzulegen und diesen anschließend zu
füllen:

```
> DD <- c(Sys.Date())
> for(i in 1:length(SO$DeliveryDate)) {
+   DD[i] = as.Date(SO$DeliveryDate[i], format="%Y-%m-%d")
+ }
> str(DD)
 Date[1:10], format: "2012-01-01" "2012-01-02" "2012-01-02" ...
```

Die eigentliche Lösung ist die Vermeidung der for-Schleife unter Ausnutzung von Vektoroperationen, zum Beispiel

```
> SO$DeliveryDate <- as.Date(SO$DeliveryDate, format="="%Y-%m-%d")
```

Mehr "Fallen" bei der Nutzung von R findet man in Patrick Burns' Tutorial (http://www.burns-stat.com/pages/Tutor/R_inferno.pdf).
