Einführung Clusteranalyse

Karsten Lübke

Clusteranalyse

Wir werden einen *simulierten* Datensatz aus *Chapman & Feit (2015): R for Marketing Research and Analytics. Springer* analysieren (http://r-marketing.r-forge.r-project.org). Näheres dazu siehe Kapitel 5 dort.

Sie können ihn von hier als `csv`-Datei herunterladen:

```
download.file("https://goo.gl/eUm8PI", destfile = "segment.csv")
```

Das Einlesen erfolgt, sofern die Daten im Arbeitsverzeichnis liegen, wieder über:

```
segment <- read.csv2("segment.csv")
```

Ein Überblick über die Daten:

```
str(segment)
```

```
## 'data.frame':    300 obs. of  7 variables:
##  $ Alter         : num  50.2 40.7 43 40.3 41.1 ...
##  $ Geschlecht    : Factor w/ 2 levels "Frau","Mann": 2 2 1 2 1 2 1 2 1 1 ...
##  $ Einkommen     : num  51356 64411 71615 42728 71641 ...
##  $ Kinder        : int  0 3 2 1 4 2 5 1 1 0 ...
##  $ Eigenheim     : Factor w/ 2 levels "Ja","Nein": 2 2 1 2 2 1 2 2 2 2 ...
##  $ Mitgliedschaft: Factor w/ 2 levels "Ja","Nein": 2 2 2 2 2 2 2 1 1 2 2 ...
##  $ Segment       : Factor w/ 4 levels "Aufsteiger","Gemischte Vorstadt",..: 2 2 2 2 2 2 2 2 2 2 .
```

```
head(segment)
```

```
##      Alter Geschlecht Einkommen Kinder Eigenheim Mitgliedschaft
## 1 50.16825       Mann  51355.75      0      Nein           Nein
## 2 40.67330       Mann  64411.10      3      Nein           Nein
## 3 42.98939       Frau  71614.94      2        Ja           Nein
## 4 40.31963       Mann  42727.96      1      Nein           Nein
## 5 41.08368       Frau  71640.62      4      Nein           Nein
## 6 40.17045       Mann  60325.41      2        Ja           Nein
##              Segment
## 1 Gemischte Vorstadt
## 2 Gemischte Vorstadt
## 3 Gemischte Vorstadt
## 4 Gemischte Vorstadt
## 5 Gemischte Vorstadt
## 6 Gemischte Vorstadt
```

Zur Unterstützung der Analyse wird (wieder) `mosaic` verwendet:

```
library(mosaic)
```

Das Ziel einer Clusteranalyse ist es, Gruppen von Beobachtungen (d. h. *Cluster*) zu finden, die innerhalb der Cluster möglichst homogen, zwischen den Clustern möglichst heterogen sind. Um die Ähnlichkeit von Beobachtungen zu bestimmen, können verschiedene Distanzmaße herangezogen werden. Für metrische Merkmale wird z. B. häufig die euklidische Metrik verwendet, d. h., Ähnlichkeit und Distanz werden auf Basis des euklidischen Abstands bestimmt. Aber auch andere Abstände wie Manhatten oder Gower sind möglich. Letztere haben den Vorteil, dass sie nicht nur für metrische Daten sondern auch für gemischte Variablentypen verwendet werden können.

Auf Basis der drei metrischen Merkmale (d. h. `Alter`, `Einkommen` und `Kinder`) ergeben sich für die ersten sechs Beobachtungen folgende Abstände:

```
dist(head(segment))
```

```
## Warning in dist(head(segment)): NAs durch Umwandlung erzeugt
```

```
##             1           2           3           4           5
## 2 19942.38233
## 3 30946.42599 11004.04804
## 4 13179.17559 33121.54360 44125.59103
## 5 30985.65446 11043.27434    39.45317 44164.81792
## 6 13701.39082  6240.99480 17245.04247 26880.54893 17284.26910
```

Sie können erkennen, dass die Beobachtungen `5` und `3` den kleinsten Abstand haben, während `5` und `4` den größten haben. Allerdings zeigen die Rohdaten auch, dass die euklidischen Abstände von der Skalierung der Variablen abhängen (`Einkommen` streut stärker als `Kinder`). Daher kann es evt. sinnvoll sein, die Variablen vor der Analyse zu standardisieren (z. B. über `scale()`). Die Funktion `daisy()` aus dem Paket `cluster` bietet hier nützliche Möglichkeiten.

```
library(cluster)

daisy(head(segment))
```

```
## Dissimilarities :
##           1         2         3         4         5
## 2 0.3073211
## 3 0.5598210 0.3901170
## 4 0.2190697 0.1836184 0.5023068
## 5 0.5157498 0.2201562 0.2416431 0.4037746
## 6 0.4014618 0.2059440 0.2389181 0.2676518 0.4261002
##
## Metric :  mixed ;  Types = I, N, I, I, N, N, N
## Number of objects : 6
```

Hierarchische Clusteranalyse

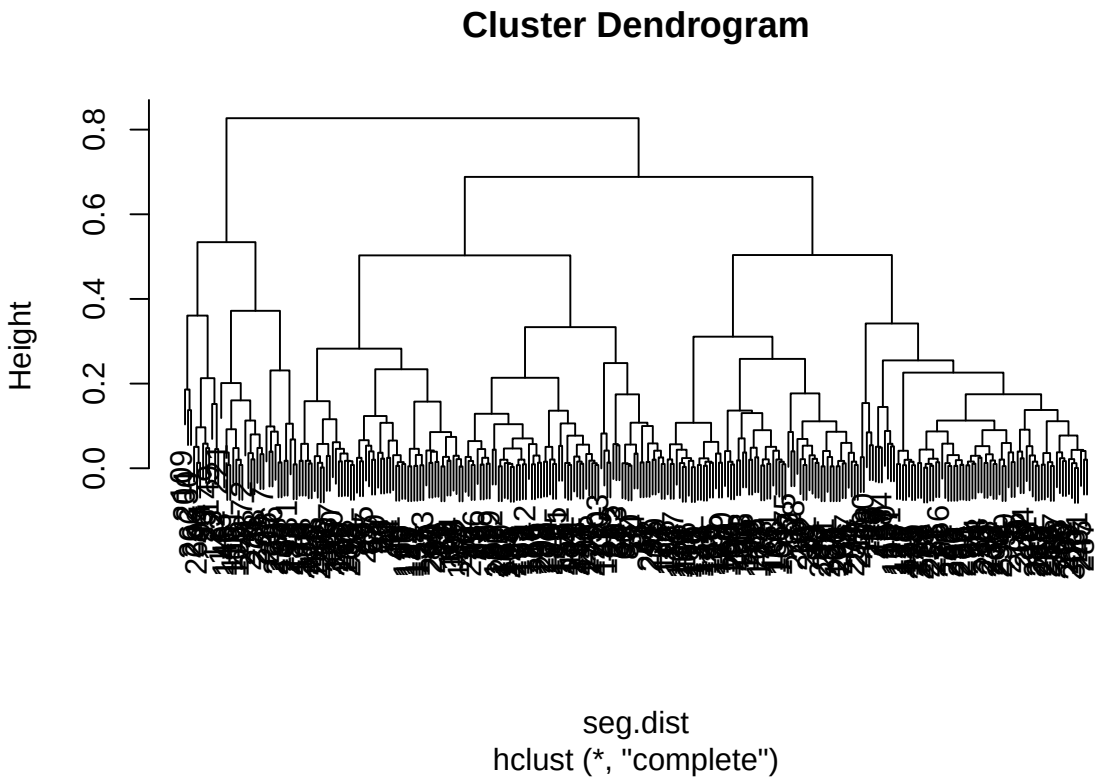
Bei hierarchischen Clusterverfahren werden Beobachtungen sukzessiv zusammengefasst (agglomerativ). Zunächst ist jede Beobachtung ein eigener Cluster, die dann je nach Ähnlichkeitsmaß zusammengefasst werden.

Fassen wir die Beobachtungen *ohne* die Segmentvariable `Segment`, Variable 7, zusammen:

```
seg.dist <- daisy(segment[,-7]) # Abstände
seg.hc <- hclust(seg.dist) # Hierarchische Clusterung
```

Das Ergebnis lässt sich schön im Dendrogramm darstellen:

```
plot(seg.hc)
```

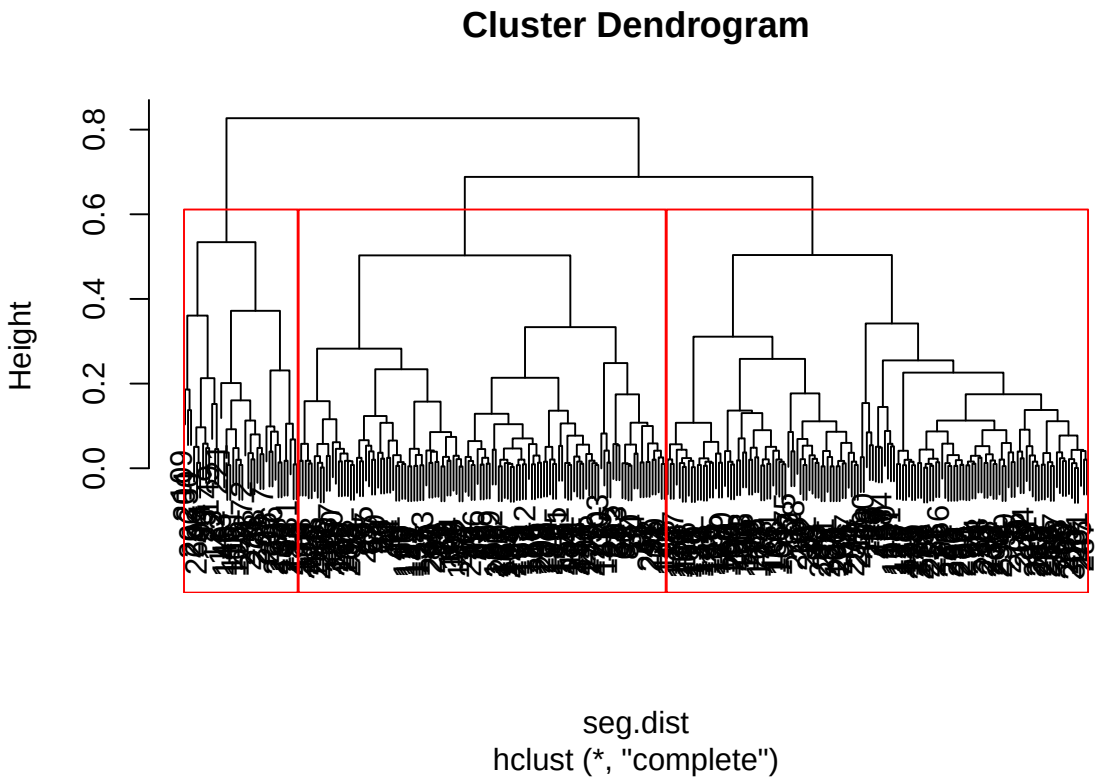


Je höher (`Height`) die Stelle ist, an der zwei Beobachtungen oder Cluster zusammengefasst werden, desto größer ist die Distanz. D. h., Beobachtungen bzw. Cluster, die unten zusammengefasst werden, sind sich ähnlich, die, die oben zusammengefasst werden unähnlich.

Hier wurde übrigens die Standardeinstellung für die Berechnung des Abstands von Clustern verwendet: Complete Linkage bedeutet, dass die Distanz zwischen zwei Clustern auf Basis des maximalen Abstands der Beobachtungen innerhalb des Clusters gebildet wird.

Es ist nicht immer einfach zu entscheiden, wie viele Cluster es gibt. In der Praxis und Literatur finden sich häufig Zahlen zwischen 3 und 10. Evt. gibt es im Dendrogramm eine Stelle, an der der Baum gut geteilt werden kann. In unserem Fall vielleicht bei einer Höhe von 0.6, da sich dann 3 Cluster ergeben:

```
plot(seg.hc)
rect.hclust(seg.hc, h=0.6, border="red")
```



Das Ergebnis, d. h. die Clusterzuordnung, kann durch den Befehl `cutree()` den Beobachtungen zugeordnet werden.

```
segment$hc.clust <- cutree(seg.hc, k=3)
```

Z. B. haben wir folgende Anzahlen für Beobachtungen je Cluster:

```
tally(~hc.clust, data=segment)
```

```
## hc.clust
##   1   2   3
## 140 122  38
```

Cluster 3 ist also mit Abstand der kleinste Cluster (mit 38 Beobachtungen).

Für den Mittelwert des Alters je Cluster gilt:

```
mean(Alter~hc.clust, data=segment)
```

```
##        1        2        3
## 38.53910 46.42365 34.48440
```

D. h., das Durchschnittsalter ist in Cluster 3 ca. 4 Jahre jünger als in 1 – und 12 Jahre jünger als in 2.

Das spiegelt sich auch im Einkommen wieder:

```
mean(Einkommen~hc.clust, data=segment)
```

```
##        1        2        3
## 49452.33 54355.46 44113.01
```

Allerdings sind die Unterschiede in der Geschlechtsverteilung eher gering:

```
tally(Geschlecht~hc.clust, data=segment, format="proportion")
```

```
##           hc.clust
## Geschlecht         1         2         3
```

```
##       Frau 0.5428571 0.5491803 0.5263158
##       Mann 0.4571429 0.4508197 0.4736842
```

k-Means Clusteranalyse

Beim k-Means Clusterverfahren handelt es sich im Gegensatz zur hierarchischen Clusteranalyse um ein partitionierendes Verfahren. Die Daten werde in k Cluster aufgeteilt – dabei muss die Anzahl der Cluster im vorhinein feststehen. Ziel ist es, dass die Quadratsumme der Abweichungen der Beobachtungen im Cluster zum Clusterzentrum minimiert wird.

Der Ablauf des Verfahrens ist wie folgt:

1. Zufällige Beobachtungen als Clusterzentrum
2. Zuordnung der Beobachtungen zum nächsten Clusterzentrum (Ähnlichkeit, z. B. über die euklidische Distanz)
3. Neuberechnung der Clusterzentren als Mittelwert der dem Cluster zugeordneten Beobachtungen

Dabei werden die Schritte 2. und 3. solange wiederholt, bis sich keine Änderung der Zuordnung mehr ergibt – oder eine maximale Anzahl an Iterationen erreicht wurde.

Hinweis: Die (robuste) Funktion `pam()` aus dem Paket `cluster` kann auch mit allgemeinen Distanzen umgehen. Außerdem für gemischte Variablentypen gut geeignet: Das Paket `clustMixType`.

Zur Vorbereitung überführen wir die nominalen Merkmale in logische, d. h. binäre Merkmale, und löschen die Segmente sowie das Ergebnis der hierarchischen Clusteranalyse:

```
segment.num <- segment %>%
  mutate(Frau = Geschlecht=="Frau") %>%
  mutate(Eigenheim = Eigenheim=="Ja") %>%
  mutate(Mitgliedschaft = Mitgliedschaft=="Ja") %>%
  select(-Geschlecht, -Segment, -hc.clust)
```

Über die Funktion `mutate()` werden Variablen im Datensatz erzeugt oder verändert. Über `select()` werden einzene Variablen ausgewählt. Die "Pfeife" `%>%` übergeben das Ergebnis der vorherigen Funktion an die folgende.

Aufgrund von (1.) hängt das Ergebnis einer k-Means Clusteranalyse vom Zufall ab. Aus Gründen der Reproduzierbarkeit sollte daher der Zufallszahlengenerator gesetzt werden. Außerdem bietet es sich an verschiedene Startkonfigurationen zu versuchen. in der Funktion `kmeans()` erfolgt dies durch die Option `nstart=`. Hier mit `k=4` Clustern:

```
set.seed(1896)

seg.k <- kmeans(segment.num, centers = 4, nstart = 10)
seg.k
```

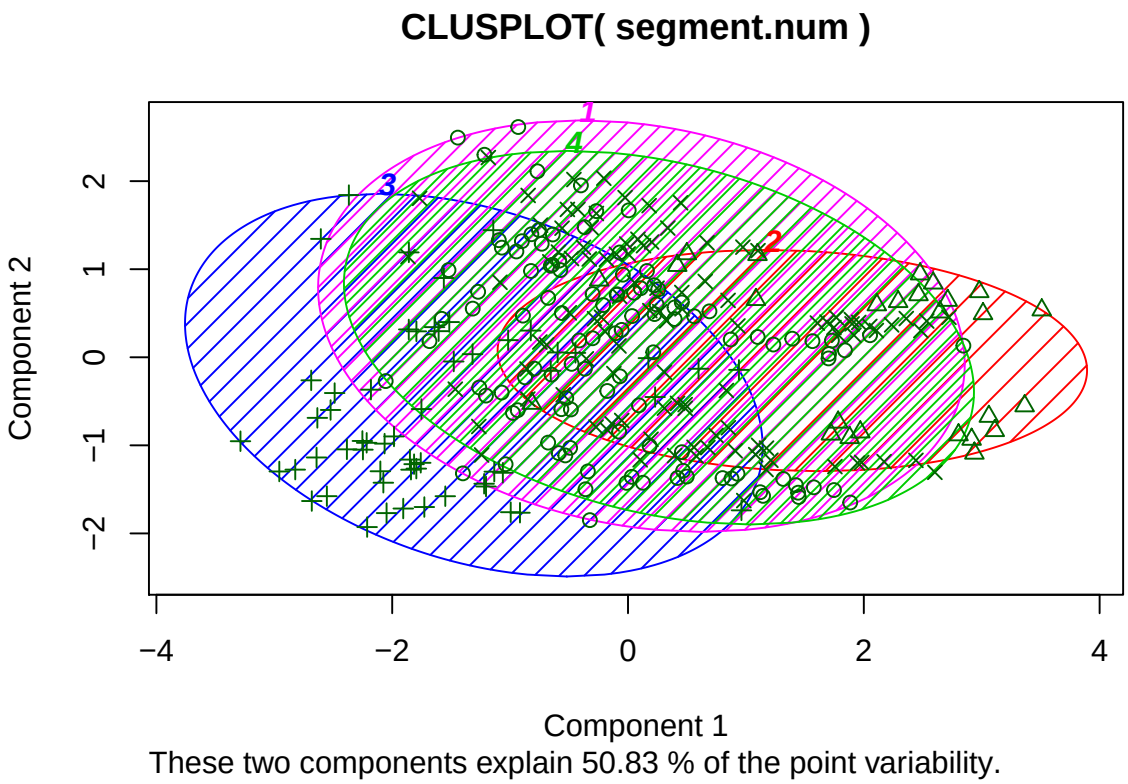
```
## K-means clustering with 4 clusters of sizes 111, 26, 58, 105
##
## Cluster means:
##      Alter Einkommen    Kinder Eigenheim Mitgliedschaft      Frau
## 1 42.91293  46049.08 1.6486486 0.5045045     0.10810811 0.5675676
## 2 56.35833  85972.56 0.3846154 0.5384615     0.03846154 0.5384615
## 3 27.01882  22607.81 1.2241379 0.2758621     0.20689655 0.4137931
## 4 43.56022  62599.99 1.5047619 0.4571429     0.12380952 0.5904762
##
## Clustering vector:
##   [1] 1 4 4 1 4 4 4 1 2 4 1 1 4 4 1 1 1 1 1 4 4 4 1 4 1 1 1 1 4 1 4 4 1 1 2
##  [36] 1 4 1 1 4 4 4 1 4 4 4 4 1 1 1 1 1 2 1 1 4 4 4 4 1 4 1 4 1 1 1 1 4 4 4
##  [71] 4 1 1 4 1 1 4 4 4 4 1 4 1 3 1 4 1 1 1 1 4 4 4 1 1 4 1 4 4 4 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 1 2 4 2 2 4 1 1 2 2 4 4 1 1 4 2 4 4 1 2 2 3 4 1 2
## [176] 2 4 2 3 4 4 4 1 1 1 1 1 1 4 3 1 4 4 4 4 1 1 1 2 4 4 1 2 4 4 1 4 2 1 2
```

```
## [211] 4 3 4 2 2 4 2 1 4 3 1 2 2 4 2 4 4 1 4 4 1 1 1 1 1 3 1 1 4 1 4 3 1 4 1
## [246] 4 1 4 1 4 4 4 4 1 1 1 4 4 1 1 1 1 1 1 4 1 1 1 1 1 2 4 4 1 4 1 1 1 1 2
## [281] 4 4 4 4 1 4 1 4 4 4 1 4 1 4 1 4 1 1 4 1
##
## Within cluster sum of squares by cluster:
## [1] 3178960729 2217438154 1689348448 2811630349
##  (between_SS / total_SS =  90.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Neben der Anzahl Beobachtungen im Cluster (z. B. 26 in Cluster 2) werden auch die Clusterzentren ausgegeben. Diese können dann direkt verglichen werden. Sie sehen z. B., dass das Durchschnittsalter in Cluster 3 mit 27 am geringsten ist. Der Anteil der Eigenheimbesitzer ist mit 54 % in Cluster 2 am höchsten.

Einen Plot der Scores auf den beiden ersten Hauptkomponenten können Sie über die Funktion `clusplot()` aus dem Paket `cluster` erhalten.

```
clusplot(segment.num, seg.k$cluster,
         color = TRUE, shade = TRUE, labels = 4)
```



Wie schon im deskriptiven Ergebnis: Die Cluster `1` und `4` unterscheiden sich (in den ersten beiden Hauptkomponenten) nicht wirklich. Vielleicht sollten dies noch zusammengefasst werden, d. h., mit `centers=3` die Analyse wiederholt werden?[1]

[1] Das Paket `NbClust`, siehe Malika Charrad, Nadia Ghazzali, Veronique Boiteau, Azam Niknafs (2014) *NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set*, Journal of Statistical Software, 61(6), 1-36. http://dx.doi.org/10.18637/jss.v061.i06, bietet viele Möglichkeiten die Anzahl der Cluster optimal zu bestimmen.

Übung: B3 Datensatz

Der B3 Datensatz *Heilemann, U. and Münch, H.J. (1996): West German Business Cycles 1963-1994: A Multivariate Discriminant Analysis. CIRET–Conference in Singapore, CIRET–Studien 50.* enthält Quartalsweise Konjunkturdaten aus (West-)Deutschland.

Er kann von https://goo.gl/0YCEHf heruntergeladen werden.

1. Wenn die Konjunkturphase `PHASEN` nicht berücksichtigt wird, wie viele Cluster könnte es geben? Ändert sich das Ergebnis, wenn die Variablen standardisiert werden?
2. Führen Sie eine k-Means Clusteranalyse mit 4 Clustern durch. Worin unterscheiden sich die gefundenen Segmente?

Literatur

- Chris Chapman, Elea McDonnell Feit (2015): *R for Marketing Research and Analytics*, Kapitel 11.3
- Reinhold Hatzinger, Kurt Hornik, Herbert Nagel (2011): *R – Einführung durch angewandte Statistik.* Kapitel 12
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013): *An Introduction to Statistical Learning – with Applications in R*, http://www-bcf.usc.edu/~gareth/ISL/, Kapitel 10.3, 10.5

Versionshinweise:

- Datum erstellt: 2017-02-08
- R Version: 3.3.2
- `mosaic` Version: 0.14.4
- `cluster` Version: 2.0.5