Autor: Hartmut Bömermann, Amt für Statistik Berlin-Brandenburg

Clusteranalyse mit der Open-Source-Software R

Vorbemerkung

R kostenlos aus dem Internet downloaden

R ist ein Programm und eine Programmiersprache für die statistische Analyse, graphische Exploration und Präsentation. Entstanden ist R als freie Implementierung der Sprache S (AT&T Bell). Seit vielen Jahren ist S-Plus als kommerzielle Implementierung am Markt vertreten. Entwickelt wird R unter der General Public License (GNU), es ist damit eine quelloffene Alternative zum kommerziellen S. Anders als die klassischen Softwarepakte wie SPSS und SAS ist R eine „Internetsoftware“ mit großer Entwickler- und Nutzergemeinschaft, einer Internetbibliothek und Foren. Als Plattform werden Windows, Linux und Mac OS X unterstützt.

R kann Daten aus Excel, SAS oder SPSS importieren. Erweiterungen in C, C++ oder Fortran sind möglich. Die R-Plattform im Web bietet fertig kompilierte Pakete für Windows, Mac OS X und Linux-Distributionen.

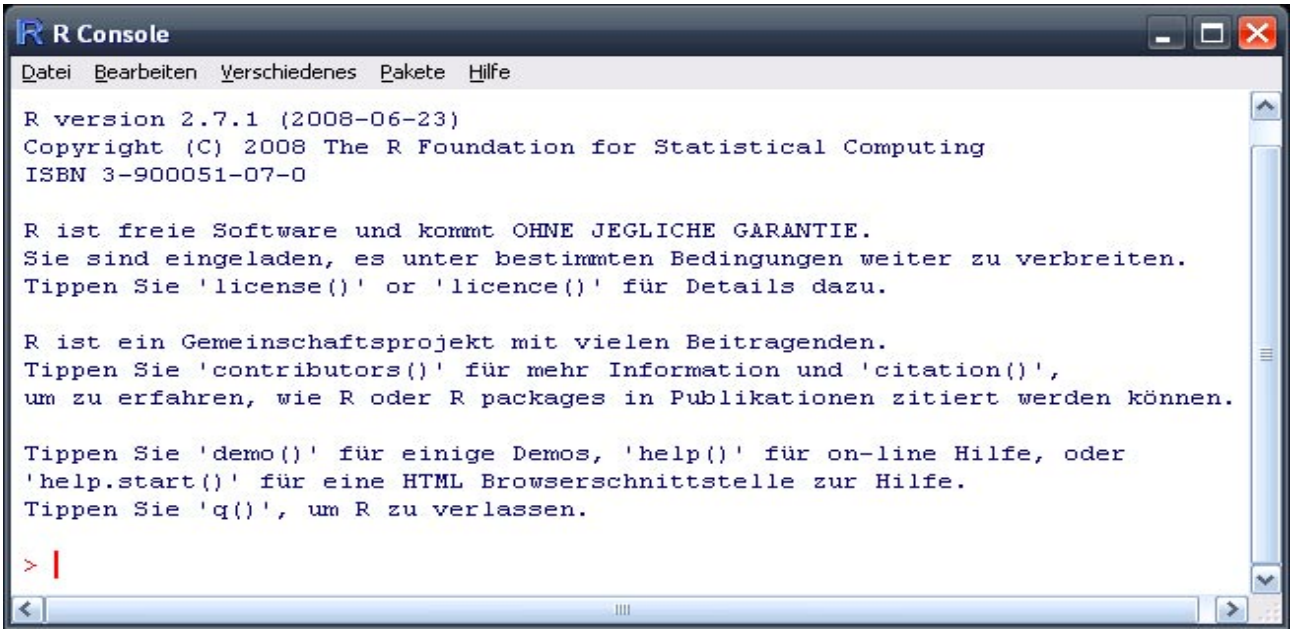
Die Anwendung kommen wohl mehrheitlich aus den Bereichen Ökonomie, Biologie und Geologie. In der Lehre wird R im „Statistiklabor“ (www.statistiklabor.de) erfolgreich eingesetzt. Im Forschungsdatenzentrum der Statistischen Ämter der Länder (www.forschungsdatenzentrum.de) steigt die Zahl der R-Anwender seit Jahren kontinuierlich an, ist aber im Vergleich zu Stata und SPSS eher klein.

R-Heimat und Installation

Das Programm und modulare Erweiterungen werden auf dem CRAN-Server zum kostenfreien Download angeboten:

www.r-project.org/

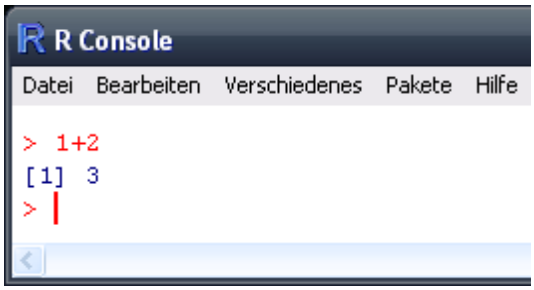
Für die folgenden Beispiele wurden für einen Windows XP-PC die Distribution R-2.7.1-win32.exe vom CRAN-Server heruntergeladen (32 MB) und danach wurde der Setup-Assistent gestartet. Nach dem Programmaufruf wird die spartanische R-Konsole RGUI geöffnet (Abb. 1). Von der R-Gemeinschaft wurden weitere Editoren entwickelt (z.B. Tinn-R, www.sciviews.org/Tinn-R), die hier nicht verwendet werden sollen.



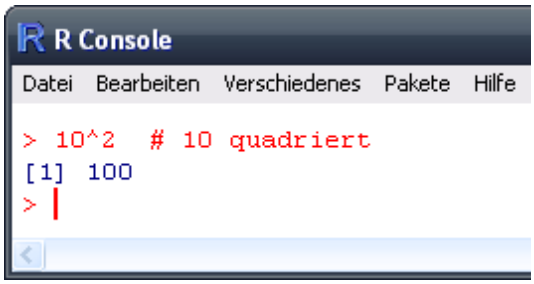
Erste Schritte: R als Taschenrechner

Vertraut machen mit R – so funktioniert's

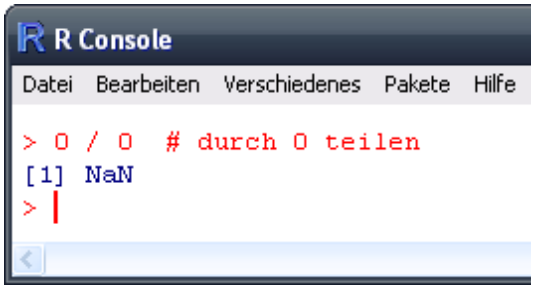
Um ein Gefühl für das Programm zu bekommen, sind erste Versuche mit R als Taschenrechner eine gute Übung.



Additionsaufgabe
1+2

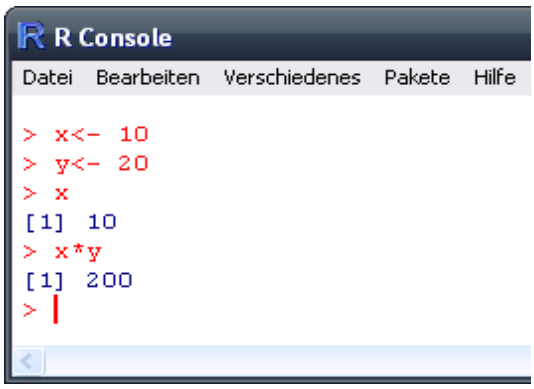


Aufgabe mit Kommentar



Division durch 0

Ausgabe NaN – Not a Number



Zuweisung eines Wertes zu einem Objekt

x=10, y=20

Die Eingabe von „x" gibt den Wert aus. „x*y" rechnet die Multiplikation aus.

Mit `help.start()` wird im Web-Browser eine Link-Seite aufgerufen, die auf Handbücher und FAQs sowie eine Suchmaschine verweist. Die Seite ist ein guter Einstieg.

Clusteranalyse mit R

Der hier kurz vorgestellte Ablauf einer Clusteranalyse gliedert sich in die Schritte:

- Daten einlesen
- deskriptive Statistiken berechnen
- Variablen standardisieren
- Clustern
- Ergebnis graphisch und kartographisch darstellen.

Die ersten Schritte zur Clusterananalyse mit R

Datengrundlage sind die Planungsräume, das ist die untere Ebene der lebensweltlich orientierten Räume (LORs), die das Berliner Stadtgebiet flächendeckend in 447 Teilräume gliedern. Jedes Aggregat wird über die achtstellige Variable `RAUMID` identifiziert. Als Indikatoren sollen die Variablen `p0U3` bis `pSGBIIPers` Variablen verwendet werden:

- `-Pop` Einwohner insgesamt (wird als Gewichtungsvariable benötigt)
- `-p0U3` Anteil Einwohner unter 3-Jahre
- `-p65plus` Anteil Einwohner 65 Jahre und älter
- `-pMH` Anteil Einwohner mit Migrationshintergrund
- `-pWandSaldo` Anteil Wanderungssaldo
- `-pSGBIIPers` Anteil Personen in SGBII-Bedarfsgemeinschaften an allen Einwohnern.

Erforderliche Programmstatements können zeilenweise eingeben werden oder sie werden als Skript ausgeführt. Bevor mit dem Clustern begonnen werden kann, muss das Programm initialisiert werden:

`options(digits=3)`	Setzen der Nachkommastellen
`setwd("D:/Workplace/")`	Setzen des Working Directories (kein Backslash)

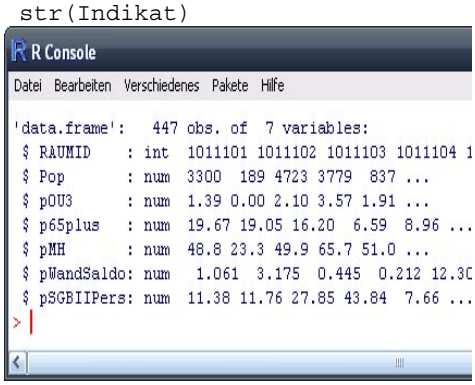
Die Daten einlesen

Im folgenden Schritt sollen die Daten eingelesen werden, die in einer Exceltabelle vorliegen. Um R zu dieser Aufgabe zu befähigen, muss das Packet RODBC nachgeladen und installiert werden. Ist der PC mit dem Internet verbunden, dann kann die Aufgabe leicht gelöst werden.

`install.packages("RODBC ", dependencies = TRUE)`	Lädt ein Erweiterungspaket vom CRAN-Server. PC muss mit dem Internet verbunden sein.
`library(RODBC)`	ODBC-Bibliothek laden.
`channel<- odbcConnectExcel("Indikatoren .xls")`	Excel-Mappe auf Objekt `channel` verweisen. Das Objekt `channel` enthält die Zugriffsparameter.
`sqlTables(channel)`	
`Indikat <- sqlQuery(channel, "select * from \"Tabelle1$\"")`	Mit SQL-Select-Statement Daten des Arbeitsblattes „Tabelle1“ auswählen und in Objekt `Indikat` speichern.
`odbcCloseAll()`	ODBC-Zugriffsobjekt wird

Datenstruktur und Daten anzeigen:

```
str(Indikat)
```

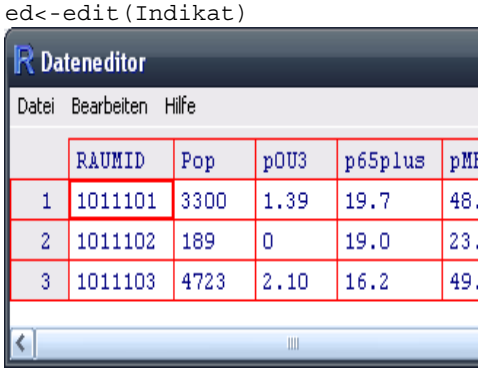


`str()` Zeigt die Struktur des Datenobjektes an: Variablen, Datentyp, Werte:
447 Fälle, 7 Variablen

- `RAUMID`	Ident des Teilraumes
- `Pop`	Einwohner insgesamt
- `p0U3`	% unter 3-Jahre
- `p65plus`	% 65 Jahre und älter
- `pMH`	% mit Migrationshintergr.
- `pWandSaldo`	% Wanderungssaldo
- `pSGBIIPers`	% Pers SGBII

```
ed<-edit(Indikat)
```

Aufruf des Dateneditors.



	RAUMID	Pop	pOU3	p65plus	pMI
1	1011101	3300	1.39	19.7	48.
2	1011102	189	0	19.0	23.
3	1011103	4723	2.10	16.2	49.

R ist objektorientiert. Die vorhandenen Objekte können mit `ls()` angezeigt werden:

```
ls()
```

Zeigt alle Objekte an: „channel“, „ed“, „Indikat“

Abgespeichert werden können die Daten (das Objekt „Indikat“) mit save() und mit load() können sie wieder geladen werden.

```
save(Indikat,
file="Indikatoren.Rdata")
```

Speichert Objekt `indikat` in Datei.

```
load("Indikatoren.RData")
```

Öffnen der Datei.

Ein einfacherer Zugriff auf die Variablen ist möglich, wenn die Daten zugeordnet werden:

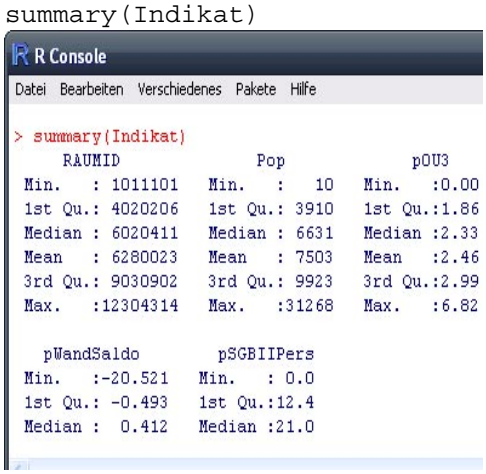
```
attach(Indikat)
```

Ordnet das Objekt zu. Vereinfacht das Ansprechen der Variablen.

Einige deskriptive Angaben.

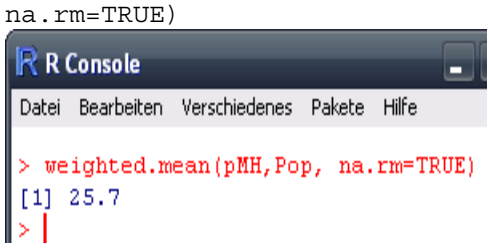
```
length(Pop)
summary(Indikat)
```

Anzahl der Fälle
Deskripte Statistiken aller Variablen (Min, Max, 1. und 3. Quartil, Median, arith. Mittelwert (ungewichtet).




```
weighted.mean(pMH,Pop,
na.rm=TRUE)
```

Mit der Variable (dem Vektor) `Pop` gewichteter Mittelwert der Variable `pMH`. Fehlende Werte werden ausgeschlossen `(=TRUE)`. Der gewichtete Mittelwert beträgt 25,7% Einwohner mit MH in Berlin.



Berechnung der Korrelationsmatrix der Variablen

Die Korrelationsmatrix der Variablen lässt sich mit `cor()` berechnen.
Es zeigt sich, das die Variablen voneinander nicht unabhängig sind, sondern teilweise mittelstark korrelieren (`p0U3 mit p65plus` und `pMH` mit `pSGBIIPers`; siehe Abbildung 2). Die zweite Altersvariable `p65plus` könnte daher ausgeschlossen werden. Auf die Reduktion des Merkmalsraumes auf die Hauptkomponenten soll verzichtet werden; die Ähnlichkeit der Teilräume soll auf Ebene der manifesten Variablen untersucht werden. Allerdings sollen die Variablen standardisiert werden, um den Einfluss unterschiedlicher Varianzen auf das Gewicht der Variablen bei der Berechnung der Ähnlichkeit/Unähnlichkeit auszuschliessen.

```
cor(as.matrix(Indikat[3:7]),
use="pairwise.complete.obs")
```

Berechnung der (ungewichteten) Korrelationsmatrix für die Variablen 3 bis 7. Korrelationen werden paarweise errechnet, d.h. fehlende Werte bei einer Variablen führen nicht zum Fallausschluss.

Abbildung 2: Korrelationsmatrix

```
> cor(as.matrix(Indikat[3:7]), use="pairwise.complete.ob
             p0U3 p65plus    pMH pWandSaldo pSGBIIPers
p0U3        1.000 -0.5347  0.330     0.1588      0.451
p65plus    -0.535  1.0000 -0.347    -0.0228     -0.285
pMH         0.330 -0.3474  1.000    -0.1106      0.544
pWandSaldo  0.159 -0.0228 -0.111     1.0000     -0.173
pSGBIIPers  0.451 -0.2846  0.544    -0.1729      1.000
```

Hierarchisches Clustern

Hierarchisch geclustert wird mit hclust().

```
HClust.1 <-
hclust(dist(scale(model.matri
x(~-1 +
as.matrix(Indikat[3:7])^2,
Indikat))), method= "ward")
```

Erstellt keine Clusterlösung und speichert das Ergebnis der Clusteranalyse in das Objekt `HClust.1`. Die Variablen 3 bis 7 werden einbezogen. Durch den scale-Parameter werden die Variablen standardisiert.

```
print(HClust.1$method)
```

Gibt die verwendete Methode aus.

Graphisch dargestellt werden kann das Dendrogramm mit `plot()`.

```
require(graphics)
```

Lädt Grafik-Bibliothek

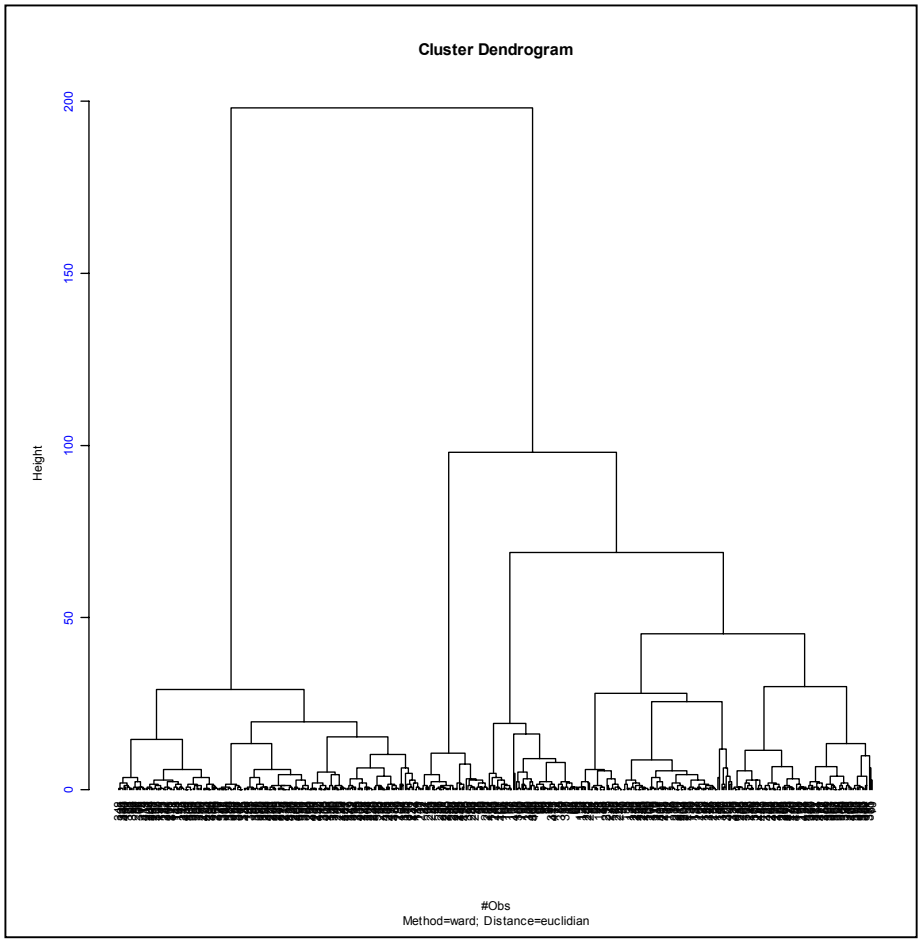
```
par(col.axis="blue", cex=0.6)
```

Setzt Grafikparameter

```
plot(HClust.1, main= "Cluster
Dendrogram", xlab= "#Obs",
sub="Method=ward;
Distance=euclidian", hang=-1)
```

Zeichnet ein Dendrogramm

Abbildung 3: Dendrogramm



Informationen über das Ergebnisobjekt

Liste 1: Struktur Clusterobjekt

```
str(HClust.1)
```

Gibt die Struktur des Objektes `HClust.1` aus (*Liste 1*).

```
List of 7
 $ merge      : int [1:445, 1:2] -381 -418 -378 -76 -48 -209 -213 -359 -138 -
349 ...
 $ height     : num [1:445] 0.0836 0.1424 0.1632 0.1709 0.2139 ...
 $ order      : int [1:446] 347 324 343 192 409 322 91 98 321 99 ...
 $ labels     : chr [1:446] "1" "2" "3" "4" ...
 $ method     : chr "ward"
 $ call       : language hclust(d = dist(scale(model.matrix(~-1 +
as.matrix(Indikat[3:7])^2,      Indikat))), method = "ward")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
> str(HClust.1)"
```

Clusteranzahl

```
plot(HClust.1$height,
type="s")
```

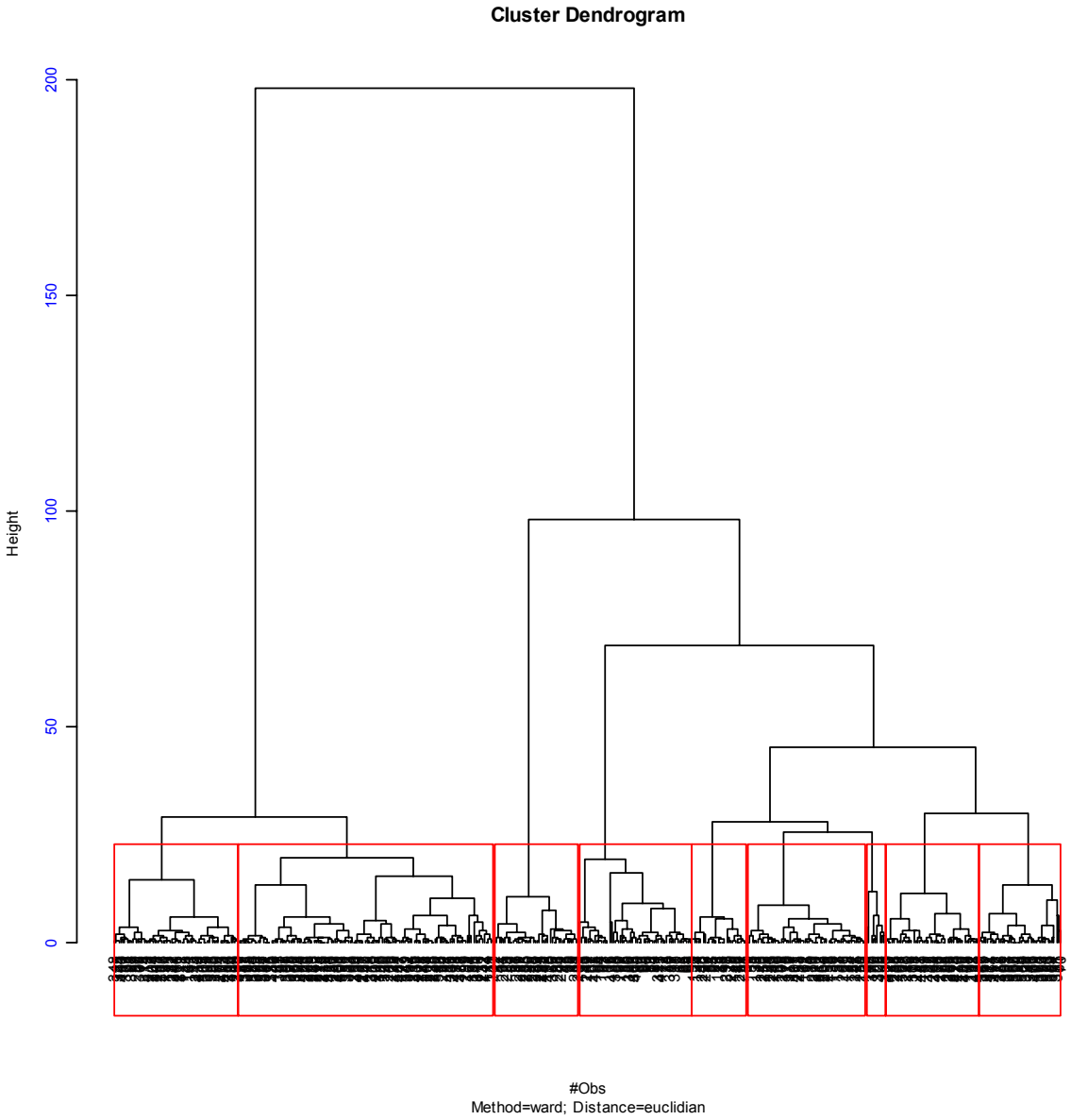
Plot des verwendeten Clusterkriteriums der Verschmelzung, bei Ward Fehlerquadrate.

Abbildung 4: Dendrogramm für 9-Clusterlösung

Dendogramm-Ausschnitte darstellen:

```
nclust=9
rect.hclust(HClust.1, k=nclust)
```

Darstellung eines Ausschnittes.



R ermöglicht die Kartierung der gefundenen Cluster. Die Geometrien können als Shape- oder MapInfo-Dateien eingelesen werden.

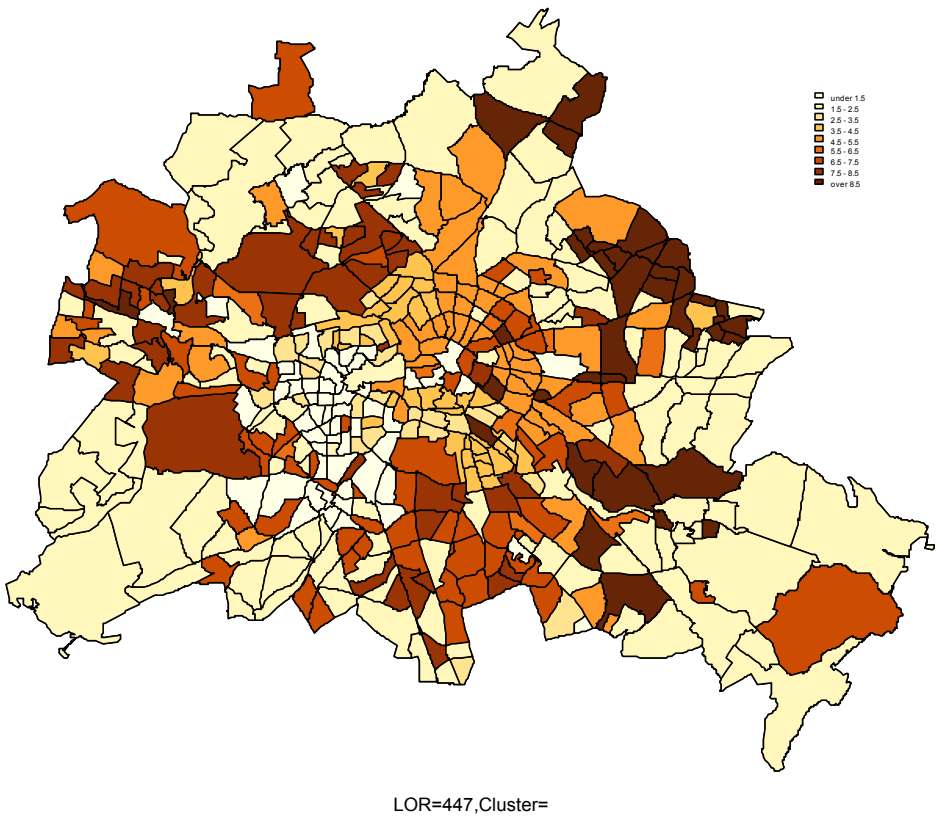
```
install.packages("rgdal", dependencies = TRUE)
install.packages("RColorBrewer", dependencies = TRUE)
install.packages("classInt", dependencies = TRUE)
library(rgdal)
library(RColorBrewer)
library(classInt)
```

```
setwd("D:/Buero/Statistik/AfS/Kommunalstat/Workplace/Cluster-AG/MapInfo/")
lor <- readOGR("LOR_PLR.TAB", "LOR_PLR")
nclust=9
clusternum <- cutree(HClust.1, k=nclust)
plotvar <- clusternum
plotclr <- brewer.pal(nclust,"YlOrBr")
```

Blues, BuPu, PuOr BuGn GnBu Greens Greys Oranges OrRd PuBu PuBuGn PuRd Purples RdPu Reds YlGn YlGnBu YlOrBr YlOrRd

```
colnum <- classIntervals(plotvar, nclust, style="fixed", fixedBreaks = seq(1, nclust, by=1),all.inside=T)
colcode <- findColours(colnum, plotclr)
plot(lor, xlim=c(5000,48000), ylim=c(4000,35000))
plot(lor, col=colcode, add=T)
title("Clusterlösung", sub="LOR=447, Cluster=")
legend(43000,35000, legend=names(attr(colcode, "table")), fill=attr(colcode, "palette"), cex=0.6, bty="n")
```

Abbildung 5:
Kartierte 9-Clusterlösung



Fazit

Vorteilhaft an R sind meines Erachtens folgende Punkte:

- Programm ist frei verfügbar
- breites statistisches Instrumentarium
- ständige Weiterentwicklung
- vielfältige Grafikmöglichkeiten
- MapInfo-, Shape-Geometrien einlesbar
- Steuerung über Syntax (leichte Wiederholbarkeit von Läufen)
- grosse Nutzer- und Entwicklergruppe
- vielfältige Hilfestellungen.

Nachteile sind:

- Programm ist nicht intuitiv
- nur rudimentäre graphische Editoren (ebenfalls GNU)
- vergleichsweise hoher Einarbeitungsaufwand.

Nutzer, die es gewohnt sind mit Syntax zu arbeiten, werden sich schneller einarbeiten können als diejenigen, die mit dem Programm über Menüs kommunizieren. Wenn aber die ersten Rezepte stehen, geht es deutlich leichter weiter. Neben dem unschlagbaren Kostenargument sprechen für R die Wiederholbarkeit von Aufgaben, die ausgefeilten graphischen Möglichkeiten sowie die Integration kartographischer Darstellungen in die statistische Analyse.

Autor:
Hartmut Bömermann
Amt für Statistik Berlin-Brandenburg
Alt-Friedrichsfelde 60
10306 Berlin
Tel.: 030 – 9021 3685
E-mail: hartmut.boemermann@statistik-bbb.de