Übungszettel Clusteranalyse

M.Psy.205, Dozent: Peter Zezula

Peter Zezula

Links

Übungszettel als PDF-Datei zum Drucken

Übungszettel mit Lösungen

Lösungszettel als PDF-Datei zum Drucken

Der gesamte Übugszettel als .Rmd-Datei (Zum Downloaden: Rechtsklick > Speichern unter...)

Hinweise zur Bearbeitung

1. Bitte beantworten Sie die Fragen in einer .Rmd Datei. Sie können Sie über `Datei > Neue Datei > R Markdown...` eine neue R Markdown Datei erstellen. Den Text unter dem *Setup Chunk* (ab Zeile 11) können Sie löschen. Unter diesem Link können Sie auch unsere Vorlage-Datei herunterladen (Rechtsklick > Speichern unter...).
2. Informationen, die Sie für die Bearbeitung benötigen, finden Sie auf der Website der Veranstaltung
3. Zögern Sie nicht, im Internet nach Lösungen zu suchen. Das effektive Suchen nach Lösungen für R-Probleme im Internet ist tatsächlich eine sehr nützliche Fähigkeit, auch Profis arbeiten auf diese Weise. Die beste Anlaufstelle dafür ist der R-Bereich der Programmiererplattform Stackoverflow
4. Auf der Website von R Studio finden Sie sehr hilfreiche Übersichtszettel zu vielen verschiedenen R-bezogenen Themen. Ein guter Anfang ist der Base R Cheat Sheet

Ressourcen

Die Clusteranalyse, um die es hier geht, wird in Field et al (2012) leider nicht dargestellt. Es findet sich allerdings in Everitt (2010) ein Kapitel (Chapter 12).

Tipp der Woche

R ist objektorientiert. Alle R-Objekte haben Attributes, auf die wir zugreifen können. Bei Dataframes und Tibbles haben wir häufig den `$` Operator benutzt. Der funktioniert nicht nur beim Zugriff auf Spalten von Datentabellen, sondern universell. Auch die Ergebnisse von Funktionen, also Analysen, sind Objekte (Ergebnisobjekte). Wir können Sie speichern und weiter benutzen. `res_obj <- lm(mpg ~ hp, data=mtcars)` Eine Auflistung der Attribute erhalten wir über die Befehle `attributes()` oder detaillierter über `str()`. Beide Befehle akzeptieren beliebige Objekte als Parameter. Im Beispiel also `attributes(res_obj)` oder `str(res_obj)`. Komfortablen Zugriff haben wir auch über den Aufklapp-Pfeil des Ergebnisobjekts im RStudio-Environment. Die Summaries benutzen ein Ergebnisobjekt als Quelle und bereiten den Inhalt für eine Ausgabe auf. Auch die Summaries können wir speichern und auf deren aufbereitete Attribute zugreifen, da auch Summaries Objekte mit Attributen sind. Wir können die Attribute direkt weiter benutzen, beispielsweise die Koeffizienten `res_obj$coefficients`. Es gibt zahlreiche Zugriffsmöglichkeiten.

Beispiel

```
res_obj <- lm(mpg ~ hp, data=mtcars)
# get the coefficients of the adapted model
res_obj$coefficients
# slicing works
res_obj$coefficients[1]
summary(res_obj)
# summaries are objects too
s_res_obj <- summary(res_obj)
# and we can access their  attributes
s_res_obj$adj.r.squared
```

1) Daten einlesen, allgemeines zum Thema

Clusteranalysen fassen Beobachtungen zu Gruppen zusammen, wobei meist mehrere Variablen miteinander kombiniert werden, um die Beobachtungen Gruppen zuzuordnen. Basis für die Zusammenfassung sind, je nach Verfahren, verschieden Ähnlichkeits- und Distanzmaße sowohl der einzelnen Beobachtungen, als auch der Gruppen, die auf Basis der benutzten Erklärvariablen gebildet werden. Es handelt sich also in der Regel um explorative Verfahren.

Besonderes Interesse finden Clusteranalysen in der Marktforschung. Daher stammt auch das Beispiel hier aus dem Bereich der Kundenforschung.

In diesem Sheet stellen wir die Befehle `stats::kmeans()` und mehrere Befehle aus der `library(mclust)` vor. Damit die Befehle gefunden werden, müssen die entsprechenden Packages geladen sein, oder wir müssen den direkten Aufruf benutzen, z. b. `mclust::densityMclust(...)`.

Der Original-Datensatz stammt von https://www.kaggle.com/dev0914sharma/customer-clustering]

The dataset consists of information about the purchasing behavior of 2,000 individuals from a given area when entering a physical 'FMCG' store. All data has been collected through the loyalty cards they use at checkout. The data has been preprocessed and there are no missing values. In addition, the volume of the dataset has been restricted and anonymised to protect the privacy of the customers.

Variable Data type Range Description

- ID numerical Integer Shows a unique identificator of a customer.
- Sex categorical {0,1} Biological sex (gender) of a customer. In this dataset there are only 2 different options.
  - 0 male
- 1 female
- Marital status categorical {0,1} Marital status of a customer.
  - 0 single
  - 1 non-single (divorced / separated / married / widowed)
- Age numerical Integer The age of the customer in years, calculated as current year minus the year of birth of the customer at the time of creation of the dataset

  - 18 Min value (the lowest age observed in the dataset)
  - 76 Max value (the highest age observed in the dataset)
- Education categorical {0,1,2,3} Level of education of the customer
  - 0 other / unknown
  - 1 high school
  - 2 university
  - 3 graduate school
- Income numerical Real Self-reported annual income in US dollars of the customer.
  - 35832 Min value (the lowest income observed in the dataset)
  - 309364 Max value (the highest income observed in the dataset)
- Occupation categorical {0,1,2} Category of occupation of the customer.
  - 0 unemployed / unskilled
  - 1 skilled employee / official
  - 2 management / self-employed / highly qualified employee / officer
- Settlement size categorical {0,1,2} The size of the city that the customer lives in.
  - 0 small city
  - 1 mid-sized city
  - 2 big city

Machen Sie sich mit dem Datensatz und seiner Struktur vertraut.

Wir stellen eine Kopie des Datensatzes auch über den Link https://md.psych.bio.uni-goettingen.de/mv/data/div/segmentation_data.csv zur Verfügung.

Aufg. 1.1 Datensatz laden

Laden Sie den Datensatz `segmentation_data.csv` in R, auf den Sie über den Link https://md.psych.bio.uni-goettingen.de/mv/data/div/segmentation_data.csv Zugriff haben. Nennen Sie das Datenobjekt `dd`.

```
library(tidyverse)
# we set the random seed to get equal results.
# As with `kmeans()` the combination of Windows and MacOS and Knitting produce different results, we re
set.seed(2341)
# we read from a local server for conveniance
dd <- readr::read_csv("https://md.psych.bio.uni-goettingen.de/mv/data/div/segmentation_data.csv")
```

Lösung

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   ID = col_double(),
##   Sex = col_double(),
##   `Marital status` = col_double(),
##   Age = col_double(),
##   Education = col_double(),
##   Income = col_double(),
##   Occupation = col_double(),
##   `Settlement size` = col_double()
## )
```

Aufg. 1.2 Umbenennen der Variablen

Die originalen Spaltennamen sind nicht gut handhabbar. Benennen Sie die Spalten also um. Benutzen Sie hierzu beispielsweise den Befehl `colnames()`. Ein lauffähiger Befehl wäre z. B. `colnames(dd) <- c("id", "genderf", "marital_status", "age", "education", "income", "occupation", "sett_size")`

Lösung

```
colnames(dd) <- c("id", "genderf", "marital_status", "age", "education", "income", "occupation", "sett_
# we take a look at the data structure
head(dd)
```

```
## # A tibble: 6 x 8
##          id genderf marital_status   age education income occupation sett_size
##       <dbl>   <dbl>          <dbl> <dbl>     <dbl>  <dbl>      <dbl>     <dbl>
## 1 100000001       0              0    67         2 124670          1         2
## 2 100000002       1              1    22         1 150773          1         2
## 3 100000003       0              0    49         1  89210          0         0
## 4 100000004       0              0    45         1 171565          1         1
## 5 100000005       0              0    53         1 149031          1         1
## 6 100000006       0              0    35         1 144848          0         0
```

Aufg. 1.3 Umskalieren der Variablen

Oft empfiehlt es sich, die Cluster-Variablen gleich zu skalieren, da sie sonst unterschiedlich stark in die Distanzmaße eingehen. Das kann über Variablen gleicher Skalierung passieren oder über eine Umskalierung verschieden skalierter Variablen. Die einfachste Variante ist dabei die z-Transformation, die in R sehr einfach über den Befehl `scale()` erreicht werden kann.

Erzeugen Sie z-transformierte Varianten der Variablen “genderf”, “marital_status”, “age”, “education”, “income”, “occupation”, “sett_size” und speichern Sie sie in neuen Variablen mit einem an den unstandardisierten Namen angehängtes `"_z"`.

Tipp: `scale()` erzeugt eine Matrix als Ergebnis. Um die zurück in einen Data-Frame zu wandeln, können Sie z. B. den Befehl `data.frame()` benutzen.

Hängen Sie die z-standardisierten Variablen an das existierende Datenobjekt an.

Lösung

```
dd.t <- dd %>% dplyr::select(genderf:sett_size) %>% scale() %>% data.frame
colnames(dd.t) <- c("genderf_z", "marital_status_z", "age_z", "education_z", "income_z", "occupation_z"
dd <- cbind(dd, dd.t)
# we take a look at the data structure
head(dd)
```

```
##      id genderf marital_status age education income occupation sett_size  genderf_z marital_status_z
## 1 1e+08       0              0  67         2 124670          1         2 -0.9171695        -0.992776
## 2 1e+08       1              1  22         1 150773          1         2  1.0897659         1.006773
## 3 1e+08       0              0  49         1  89210          0         0 -0.9171695        -0.992776
## 4 1e+08       0              0  45         1 171565          1         1 -0.9171695        -0.992776
## 5 1e+08       0              0  53         1 149031          1         1 -0.9171695        -0.992776
## 6 1e+08       0              0  35         1 144848          0         0 -0.9171695        -0.992776
##      income_z occupation_z sett_size_z
## 1  0.09749923    0.2967488   1.5519379
## 2  0.78245869    0.2967488   1.5519379
## 3 -0.83299391   -1.2692080  -0.9095021
## 4  1.32805410    0.2967488   0.3212179
## 5  0.73674749    0.2967488   0.3212179
## 6  0.62698289   -1.2692080  -0.9095021
```

2) Dentrogramm erstellen, Clusterzugehörigkeit speichern

Ein gängiges Vorgehen bei der Clusterung von Beobachtungen ist das Erstellen eine Dentrogramms. Dentrogramme stellen die Beobachtungen und deren sukkzessives Zusammenfassen zu Gruppen graphisch dar. Für das Erstellen eines Dentrogramms muss festgelegt werden, welches Prinzip für das Zusammenfassen benutzt werden soll. Gängig sind hier z. B. “Single Linkage”, “Complete Linkage”, “Average Linkage” etc.

Da wir in unserem Beispiel einen großen Datensatz mit 2000 Beobachtungen haben, würde das Dentrogramm sehr unübersichlich werden. Uns geht es hier hauptsächlich um eine Demonstration des Prinzips. Daher lassen wir ein Dentrogramm von einer Zufallsauswahl der Beobachtungen erstellen.

Aufg. 2.1 Sample erstellen

Erstellen Sie eine Zufallsauswahl von 50 Beobachtungen aus dem Datensatz und speichern Sie diese Auswahl in einem Datenobjekt mit Namen `dd.s` Sie können hierzu beispielsweise den Befehl `sample` benutzen. `?sample()` hilft, wenn Sie nicht wissen, wie der Befehl funktioniert (oder natürlich eine Suche im WWW). Sorgen Sie zusätzlich dafür, dass nur die z-standardisierten Spalten in dem Teildatensatz vorhanden sind.

Lösung

```
s.sel <- sample(dd$id, 50)
dd.s <- dd[dd$id %in% s.sel,] %>% dplyr::select(genderf_z:sett_size_z)
```

Aufg. 2.2 Distanzmatrix erstellen

Erstellen Sie eine Distanzmatrix der so ausgewählten Beobachtungen. Hierzu können Sie den Befehl `dist()` benutzen. Werfen Sie einen Blick auf die Distanzmatrix.

Lösung

```
dists <- dd.s %>% dist()
# some distances ...
dists[1:10]
```

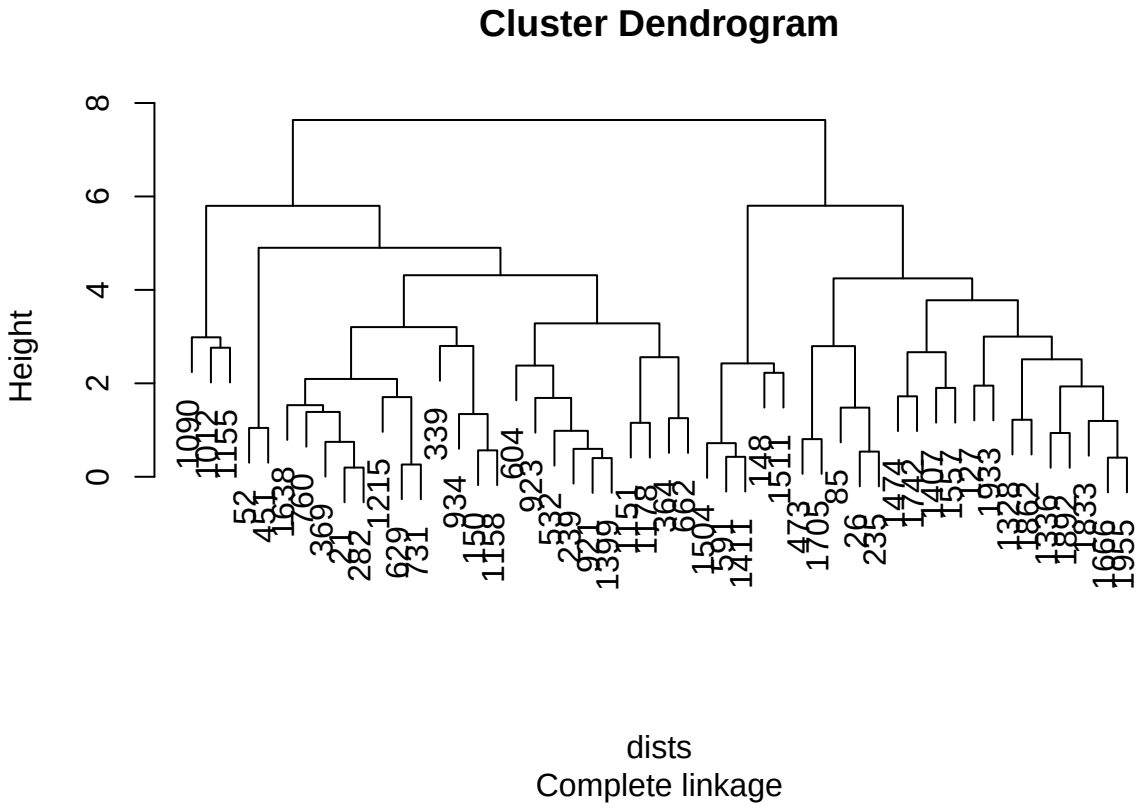
```
##  [1] 2.5549704 3.4209993 2.1251336 3.1554686 3.5273749 1.5236281 2.3142275 3.0570361 0.1973891 2.686
```

Aufg. 2.3 Dentrogramm erstellen

Erstellen Sie ein Dentrogramm für die 50 ausgewählten Beobachtungen. Wählen Sie dabei die Methode "Complete Linkage". Sie können hierzu den Befehl `plot()` auf das Ergebnis des Befehls `hclust()` anwenden, der wiederum den Parameter `method=...` verarbeiten kann. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/hclust

Lösung

```
plot(hclust(dists, method="complete"),ylab="Height", sub="Complete linkage")
```



Aufg. 2.4 Speichern von Clusterzugehörigkeiten

Speichern Sie für die ausgewählten Beobachtungen die Clusterzugehörigkeit, wenn Sie eine "Höhe" von 4 als Kriterium anwenden in einer neuen Spalte namens `ch_4`.

Speichern Sie außerdem die Clusterzugeörigkeit der ausgewählten Beobachtungen, wenn Sie genau 4 Cluster erhalten wollen. Geben Sie der Ergebnisspalte den Namen `cn_4`.

Für beides können Sie den Befehl `cutree` benutzen. Finden Sie heraus, wie das geht.

```
# we can get cluster membership for our observations for a specific method at a specific height
cutree(hclust(dists,method="complete"),h=5)
```

Lösung

```
##   21   26   52   85  127  148  150  235  239  282  339  364  369  451  473  532  591  604  629  662
##    1    2    1    2    2    3    1    2    1    1    1    1    1    1    2    1    3    1    1    1
## 1012 1090 1151 1155 1158 1178 1215 1328 1336 1399 1407 1411 1474 1504 1511 1557 1638 1666 1705 1742
##    4    4    1    4    1    1    1    2    2    1    2    3    2    3    3    2    1    2    2    2
```

```
# and store it (h is Height) h=5 results in 4 clusters, therefore the name dd.s$ch_4
dd.s$ch_4 <- cutree(hclust(dists,method="complete"),h=5)
# psych::describe(dd.s$ch_4)
# we can also store membership for a certain number of clusters
dd.s$cn_4 <- cutree(hclust(dists,method="complete"),k=4)
# we look at the resulting cluster memberships
dd.s$ch_4
```

```
##  [1] 1 2 1 2 2 3 1 2 1 1 1 1 1 1 2 1 3 1 1 1 1 1 1 1 1 4 4 1 4 1 1 1 2 2 1 2 3 2 3 3 2 1 2 2 2 2 2 2
```

```
# ... and take a look at it
head(dd.s)
```

```
##      genderf_z marital_status_z        age_z education_z    income_z occupation_z sett_size_z ch_4 cn
## 21  -0.9171695        -0.992776  1.031707902 -0.06335658 -0.05713687    0.2967488   0.3212179    1
## 26  -0.9171695        -0.992776  0.007764901 -0.06335658 -1.28698325   -1.2692080  -0.9095021    2
## 52  -0.9171695        -0.992776 -0.504206599 -1.73063500  1.55453708    1.8627056   1.5519379    1
## 85  -0.9171695        -0.992776  1.373022236 -0.06335658 -0.71501599   -1.2692080  -0.9095021    2
## 127 -0.9171695         1.006773  0.349079235 -0.06335658 -1.29241506   -1.2692080  -0.9095021    2
## 148  1.0897659         1.006773  1.287693652  1.60392184  0.18028845    0.2967488  -0.9095021    3
```

3) Herausforderung: Elbow Plot und Anzahl von Clustern

Wie bei anderen explorativen Verfahren, z. B. (eFA)[sheet_pca_fa.html], ist nicht klar, in wie viele Cluster die Beobachtungen am besten aufgeteilt werden. Hierzu hilft ein dem von der eFA bekannten Screeplot ähnliches Vorgehen. Hierzu werden die within SS (Varianz der Distanzen in den Clustern) bzw. die between SS (Varianz der Abstände zwischen den Clustern) für eine aufsteigende Anzahl von Clustern geplottet. Danach wird versucht, in dem Verlauf einen markanten Knick zu finden (Elbow). Dies kann helfen, eine geeignete Anzahl von Clustern festzulegen. Hier gibt es natürlich subjektiven Spielraum.

Bemerkung: Wenn Sie hier Schwierigkeiten haben, schauen Sie sich einfach das unten verlinkte Ergebnis an und machen Sie mit Aufgabe 4 weiter.

Aufg. 3.1 Arbeiten mit einer Schleife und Attributen von Ergebnisobjekten

Notwendiger Hintergrund: Der Befehl `kmeans()` clustert Daten und erstellt eine bestimmte Menge von Clustern. Die Menge der zu erstellenden Cluster wird bei `kmeans()` über den Parameter `centers` festgelegt.

Erstellen Sie eine Schleife, in der ein Index `k` von 1 - 10 laufen soll. Schleifen in R funktionieren nach dem Prinzip: `for (Sequenz){Befehle}`. Als `Sequenz` könnte z. B. stehen: `k in 1:10`. Hierdurch ist im Schleifenkörper `k` mit seinem jeweils aktuellen Wert bekannt. Rufen Sie im Schleifenkörper den Befehl `kmeans()` auf und übergeben Sie dem Befehl den Parameter `centers` mit der jeweiligen Ausprägung von `k`. Als ersten Parameter erwartet `kmeans()` ein Datenobjekt. Dieses Datenobjekt soll `dd` sein, allerdings nur die Spalten `dd$genderf` bis `dd$sett_size`. Benutzen Sie wahlweise den Befehl `%>%` oder die positionale Übergabe. Lassen Sie sich in der Loop den Parameter `tot.withinss` ausgeben.

```
set.seed(2341)
for (k in 1:10) {
  model <- dd %>% dplyr::select(genderf_z:sett_size_z) %>% kmeans(centers=k, iter.max=1000, nstart=1000)
  cat(model$tot.withinss)
  cat('; ')
}
```

Lösung

```
## 13993; 10509.09; 8626.587; 7166.286; 6399.869; 5828.041; 5367.033; 4957.137; 4571.29; 4269.013;
```

Aufg. 3.2 Erstellen eines Ergebnis-Datenobjekts

Erstellen Sie einen Dataframe, in dem als Zeile die Kernergebnisse der oben ausgeführten `kmeans()` Analysen stehen. Dies sollen als Spalten sein: - k (Anzahl der Cluster) - tot.withinss (gibt es als Property des Ergebnisobjekts von `kmeans()`) - betweenss (gibt es als Property des Ergebnisobjekts von `kmeans()`) - totss (gibt es als Property des Ergebnisobjekts von `kmeans()`)

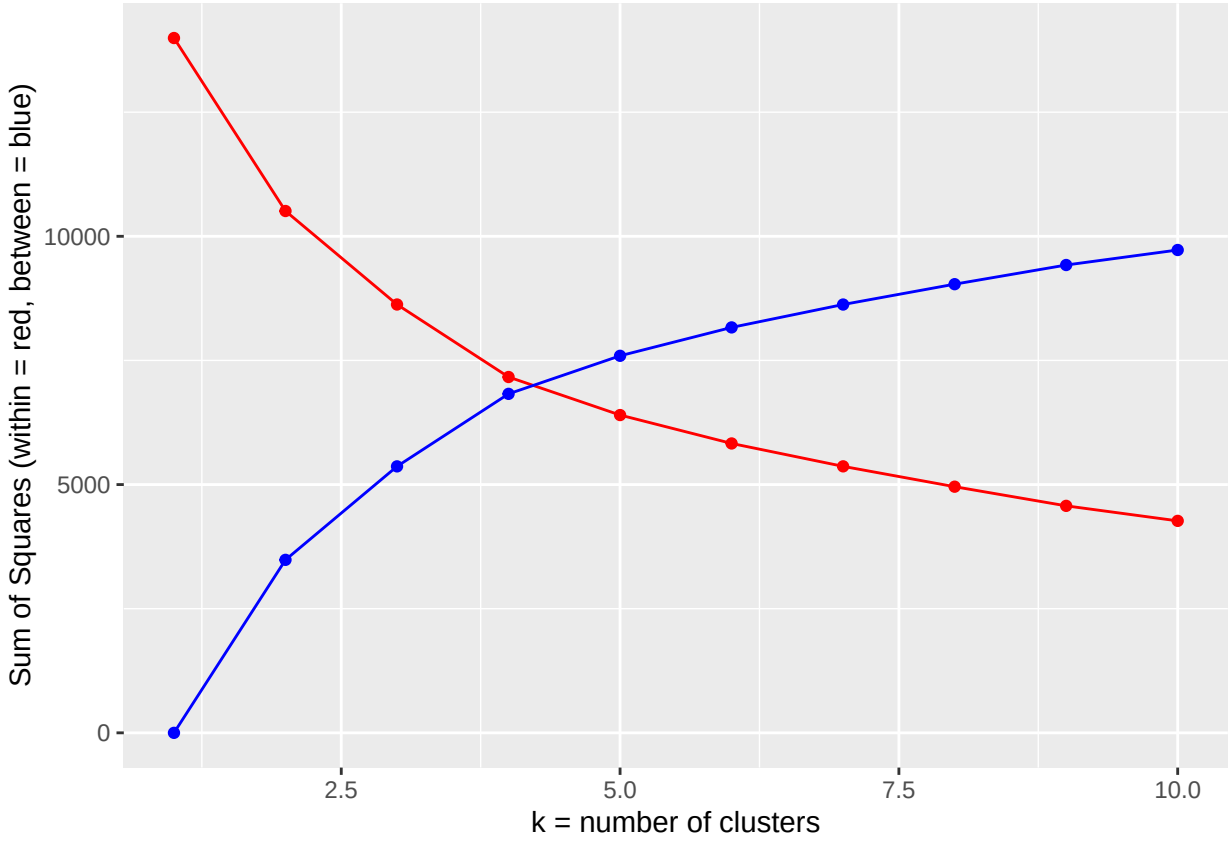
```
set.seed(2341)
criteria <- data.frame()
nk <- 1:10
for (k in nk) {
  model <- dd %>% dplyr::select(genderf_z:sett_size_z) %>% kmeans(centers=k, iter.max=1000, nstart=1000)
  criteria <- rbind(criteria,c(k, model$tot.withinss, model$betweenss, model$totss))
}
# renaming columns
names(criteria) <- c("k","tot.withinss","betweenss","totalss")
```

Lösung

Aufg. 3.3 Erstellen eines Elbow Plots

Erstellen Sie eine Grafik, in der auf der x-Achse die Anzahl der Cluster und auf der y-Achse die `tot.withinss` für die jeweilige Anzahl der Cluster grafisch dargestellt werden.

```
#scree plot
ggplot(criteria, aes(x=k)) +
  geom_point(aes(y=tot.withinss),color="red") +
  geom_line(aes(y=tot.withinss),color="red") +
  geom_point(aes(y=betweenss),color="blue") +
  geom_line(aes(y=betweenss),color="blue") +
  xlab("k = number of clusters") + ylab("Sum of Squares (within = red, between = blue)")
```



Lösung

Aufg. 3.4 Grafikbasierte Entscheidung

Sollten Sie Schwierigkeiten bei der Umsetzung von Aufg. 3.1 - 3.3 haben, finden Sie die fertige Grafik hier.

Entscheiden Sie sich auf Basis der erstellten Grafik für eine Ihrer Ansicht nach adäquaten Anzahl von Clustern.

Lösung Vermutlich kommen Sie auf 4 oder 5 Cluster.

4) Cluster mit `kmeans()` bilden

Nehmen wir an, Sie hätten sich für 4 Cluster entschieden.

Aufg. 4.1 Ansatz mit 4 Clustern

Erstellen Sie ein Modell mit 4 Clustern. Benutzen Sie hierbei alle Variablen des Datensatzes `dd`, bis auf die erste Spalte `dd$id` zur Bildung der Cluster. Benutzen Sie hierfür den Befehl `kmeans()` Speichern Sie das Modell als Ergebnisobjekt unter dem Namen `model` Schauen Sie sich an, was Ihnen das Ergebnisobjekt so alles zu bieten hat. Sie können hierfür den Befehl `attributes()` nutzen, der Ihnen die Namen aller Attribute ausgibt, die ein Objekt hat.

```
library(scales) # to manage colors via hue_pal()
# k-means 5 Clusters and use the scores
set.seed(2341)
n_clusters = 4
model <- dd %>% dplyr::select(genderf_z:sett_size_z) %>% kmeans(centers=n_clusters, iter.max=1000, nsta
attributes(model)
```

Lösung

```
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
## [9] "ifault"
##
## $class
## [1] "kmeans"
```

Aufg. 4.2 Speichern der Clusterzuordnung

Speichern Sie die Clusterzugehörigkeit der Beobachtungen in einer Spalte mit Namen `dd$kmc_4`

Lassen Sie sich die Menge der Beobachtungen ausgeben, die den jeweiligen Clustern angehören.

```
# if we want to use the clusters furtheron f. e. for some analyses, we store the cluster assignment
dd$kmc_4 <- model$cluster
# or even better
dd$kmc_4 <- factor(model$cluster)
table(dd$kmc_4)
```

Lösung

```
##
##   1   2   3   4
## 705 263 570 462
```

Aufg. 4.2 Inspektion der Clusterzentren

Die Clusterzentren sind die Mittelwerte der geclusterten Variablen für das jeweilige Cluster. Lassen Sie sich die Clusterzentren ausgeben.

Inspizieren Sie die Clusterzentren - was fällt Ihnen auf?

```
model$centers
```

Lösung

```
##     genderf_z marital_status_z       age_z education_z   income_z occupation_z sett_size_z
## 1  0.79655407        1.0011004 -0.59268205  0.05016025 -0.3987344   -0.2763247  -0.3892828
## 2  0.09011369        0.3909422  1.68902999  1.81946354  0.9809802    0.4991919   0.4569247
## 3 -0.85731349       -0.6454860 -0.02337255 -0.50796416  0.5317359    0.7225792   0.9646469
## 4 -0.20909486       -0.9538238 -0.02825041 -0.48558943 -0.6060162   -0.7540014  -0.8562241
```

```
# these are the standardized means
# to get the original means of the clusters we can:
# dd %>% dplyr::group_by(kmc_4) %>% dplyr::select(genderf:sett_size) %>% dplyr::summarize_all(mean)
( m_u <- dd %>% dplyr::group_by(kmc_4) %>% dplyr::select(genderf:sett_size) %>% dplyr::summarize_all(mea
```

```
## Adding missing grouping variables: `kmc_4`
```

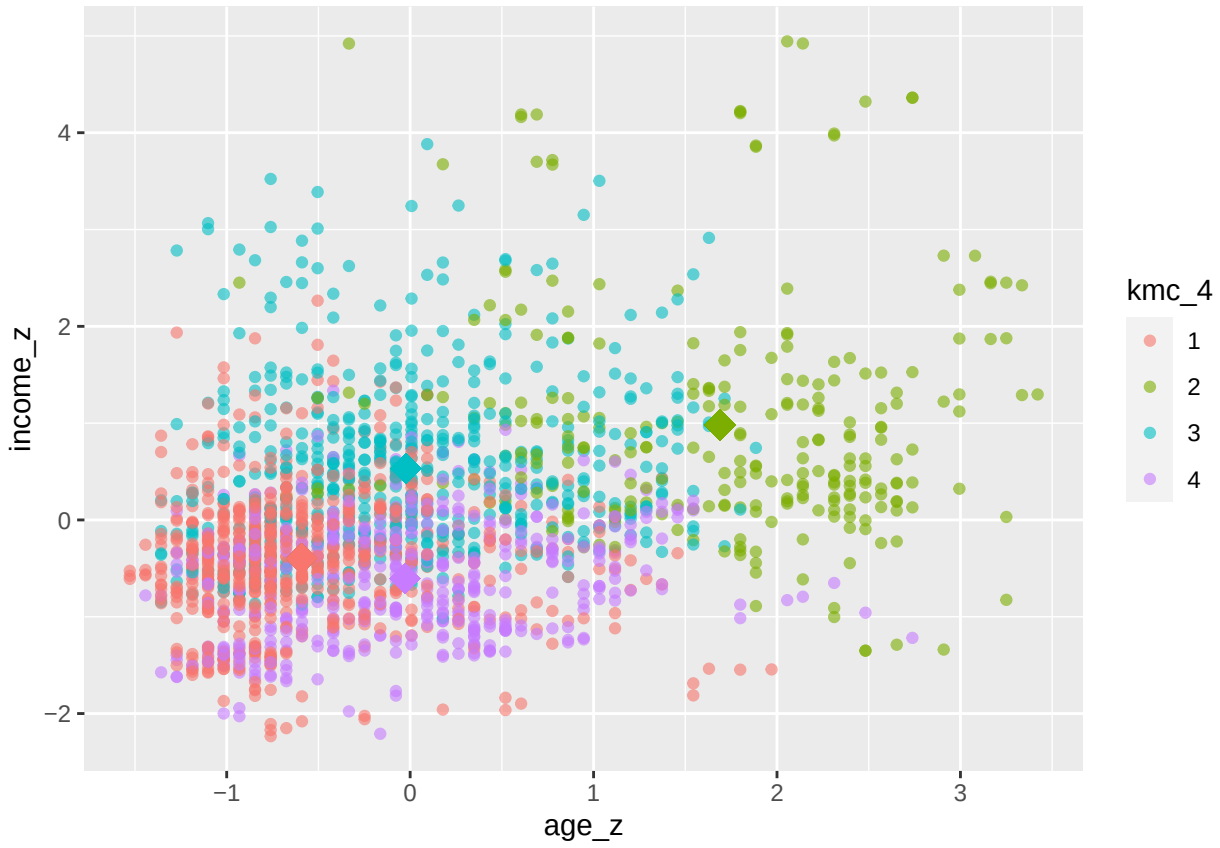
```
## # A tibble: 4 x 8
##   kmc_4 genderf marital_status   age education  income occupation sett_size
##   <fct>   <dbl>          <dbl> <dbl>     <dbl>   <dbl>      <dbl>     <dbl>
## 1 1       0.854          0.997   29.0     1.07  105759.      0.634    0.423
## 2 2       0.502          0.692   55.7     2.13  158338.      1.13     1.11
## 3 3       0.0298         0.174   35.6     0.733 141218.      1.27     1.52
## 4 4       0.353          0.0195  35.6     0.747  97860.      0.329    0.0433
```

Wir könnten z. B. ein Cluster (meist Nr 2) als eine besondere Untergruppe identifizieren. Es handelt sich überwiegend um eher ältere, verheiratete etwa gleich viele Männer oder Frauen, die sehr gut gebildet sind und ein vergleichsweise hohes Einkommen in einer guten Position haben.

Aufg. 4.4 Visualisierung der Clusterzuordnung

Visualisieren Sie die gebildeten Cluster in einem Scatterplot, in dem das Alter gegen das Einkommen geplottet ist.

```
# we look at clusters in a scatterplot of age and income
dd %>% ggplot(aes(x=age_z,y=income_z, color=kmc_4)) +
  geom_point(alpha=.6) +#plotting alll the points
  #plotting the centroids
  geom_point(aes(x=model$centers[1,"age_z"],y=model$centers[1,"income_z"]),color=hue_pal()(n_clusters)[
  geom_point(aes(x=model$centers[2,"age_z"],y=model$centers[2,"income_z"]),color=hue_pal()(n_clusters)[
  geom_point(aes(x=model$centers[3,"age_z"],y=model$centers[3,"income_z"]),color=hue_pal()(n_clusters)[
  geom_point(aes(x=model$centers[4,"age_z"],y=model$centers[4,"income_z"]),color=hue_pal()(n_clusters)[
```



Lösung

```
  # geom_point(aes(x=model$centers[5,"age_z"],y=model$centers[5,"income_z"]),color=hue_pal()(n_clusters)

# we store the above calculated unstandardized means
m_u <- dd %>% dplyr::group_by(kmc_4) %>% dplyr::select(genderf:sett_size) %>% dplyr::summarize_all(mean)
```

```
## Adding missing grouping variables: `kmc_4`
```

```
# we look at clusters in a scatterplot of age and income
dd %>% ggplot(aes(x=age,y=income, color=kmc_4)) +
  geom_point(alpha=.6) +#plotting alll the points
  #plotting the centroids
  geom_point(aes(x=as.numeric(m_u[1,"age"]),y=as.numeric(m_u[1,"income"])),color=hue_pal()(n_clusters)[
  geom_point(aes(x=as.numeric(m_u[2,"age"]),y=as.numeric(m_u[2,"income"])),color=hue_pal()(n_clusters)[
  geom_point(aes(x=as.numeric(m_u[3,"age"]),y=as.numeric(m_u[3,"income"])),color=hue_pal()(n_clusters)[
  geom_point(aes(x=as.numeric(m_u[4,"age"]),y=as.numeric(m_u[4,"income"])),color=hue_pal()(n_clusters)[
```


```
  # geom_point(aes(x=as.numeric(m_u[5,"age"]),y=as.numeric(m_u[5,"income"])),color=hue_pal()(n_clusters
```

5) Rendern

Lassen Sie die Datei mit `Strg` + `Shift` + `K` (Windows) oder `Cmd` + `Shift` + `K` (Mac) rendern. Sie sollten nun eine "schön aufpolierte" Version ihrer Datei sehen. Falls das klappt: Herzlichen Glückwunsch! Ihr Code kann vollständig ohne Fehlermeldung gerendert werden. Falls nicht: Nur mut, das wird schon noch! Gehen Sie auf Fehlersuche! Ansonsten schaffen wir es ja in der Übung vielleicht gemeinsam.

6) Abschlussbemerkungen

Die Clusterung mit `kmeans()` ist stark abhängig von der Zufalls-Initialisierung der Cluster. Daher haben wir in den Beispielen oben einige Parameter hoch gesetzt. `iter.max=1000` und `nstart=1000`. Außerdem haben wir den Random Generator von R unmittelbar vor dem Aufruf von `kmeans()` initialisiert mit `set.seed(2341)`, um möglichst vergleichbare bzw. reproduzierbare Ergebnisse zu bekommen. Wegen der starken Überlappungen unserer Cluster ist ein stabile Zuordnung der Beobachtungen zu Clustern ebenfalls erschwert.

Literatur bzw. Quellen

Anmerkung: Dieser Übungszettel basiert auf Daten und Beispielen, die von der Website https://www.kaggle.com/ stammen. Sie wurden für den Zweck dieser Übung modifiziert und entsprechender R-Code generiert.

Version: 12 August, 2021 08:58