

PRACTICAL NO 5

Aim: Implement Clustering and Associated algorithms

Software: R Studio

Download R Studio 4.3.0 from the official website (<https://posit.co/download/rstudio-desktop/> Desktop - Posit) and install it

Description:

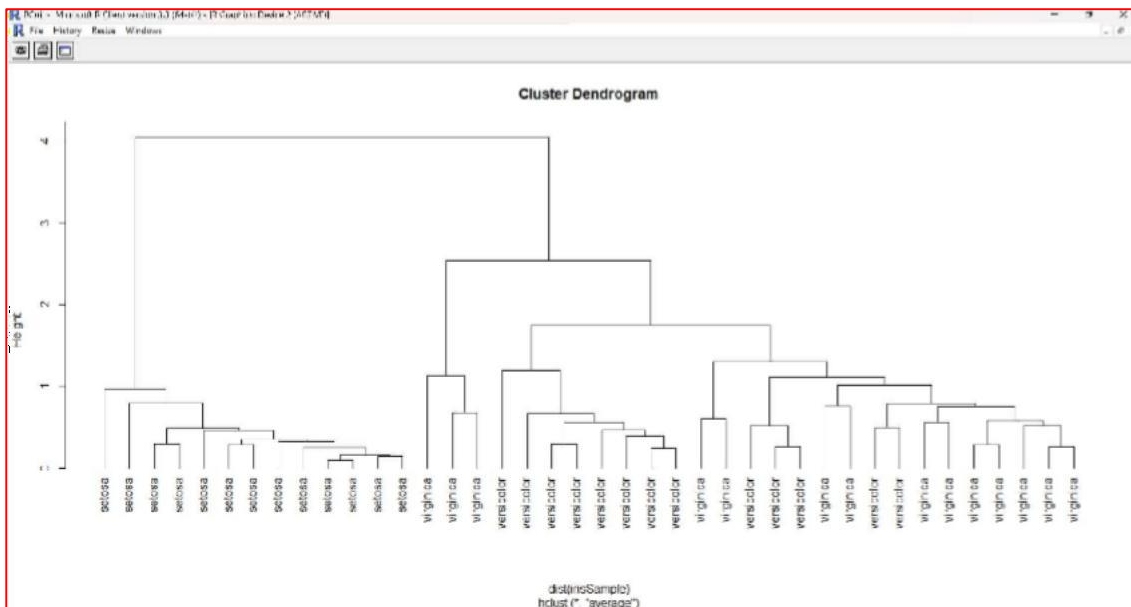
1. Data preparation
2. Assessing clustering tendency (i.e., the cluster ability of the data)
3. Defining the optimal number of clusters
4. Computing partitioning cluster analyses (e.g.: K-means, pam) or hierarchical clustering
5. Validating Clustering analyses

Code:

```
> idx<-sample(1:dim(iris)[1],40)
> irisSample<-iris[idx,]
> irisSample$Species<-NULL
> hc<-hclust(dist(irisSample),method="ave")
> plot(hc,hang=-1,labels=iris$Species[idx])
```

```
> idx<-sample(1:dim(iris)[1],40)
> irisSample<-iris[idx,]
> irisSample$Species<-NULL
> hc<-hclust(dist(irisSample),method="ave")
> plot(hc,hang=-1,labels=iris$Species[idx])
> |
```

Output:



Association Rule Mining:

Code:

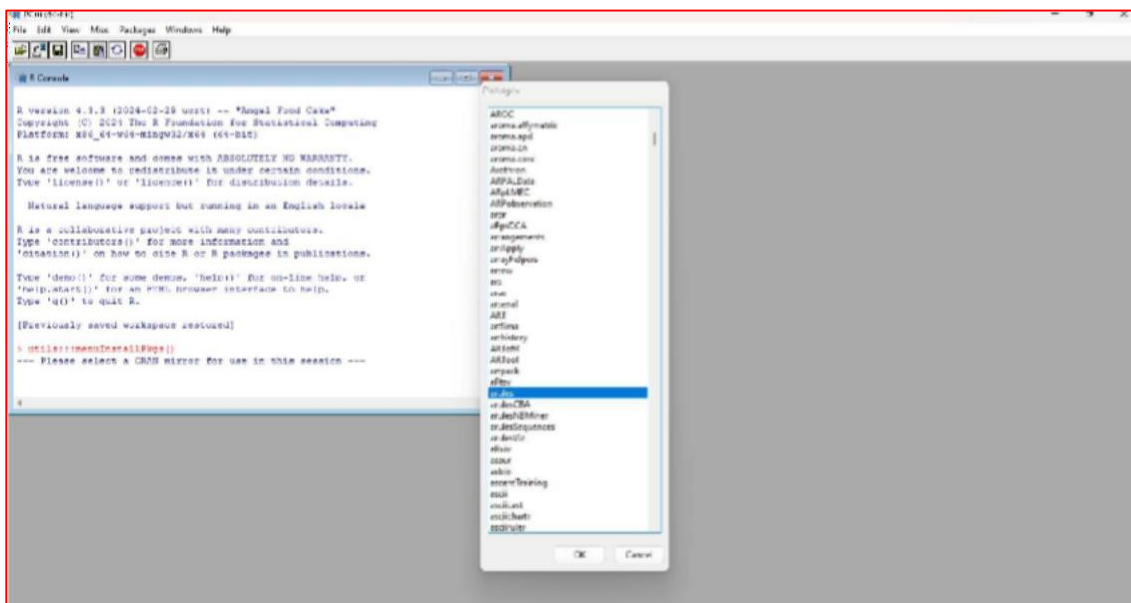
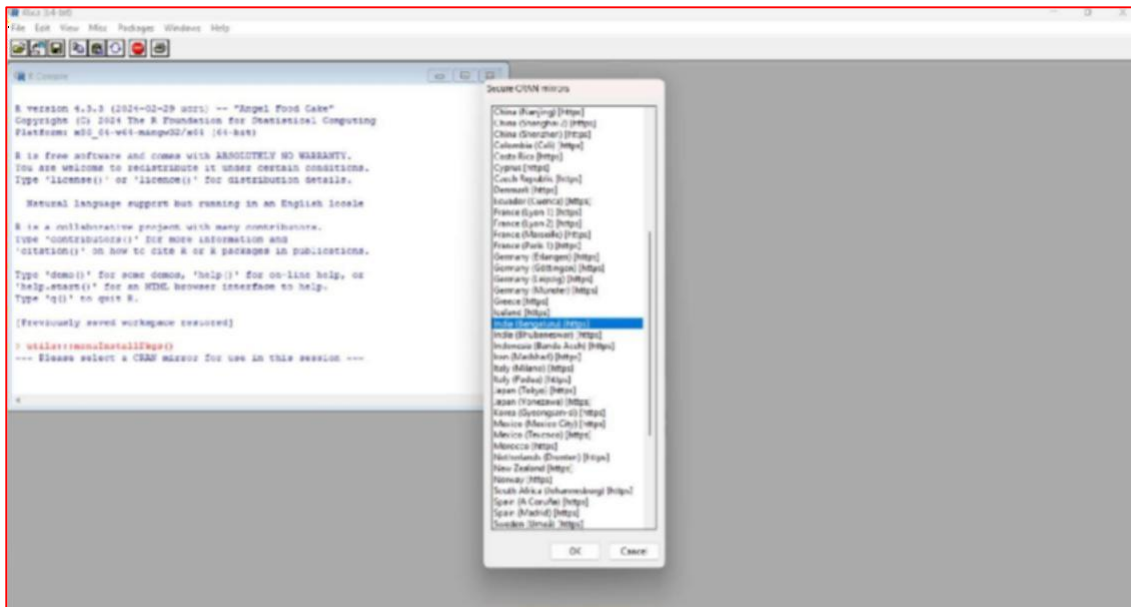
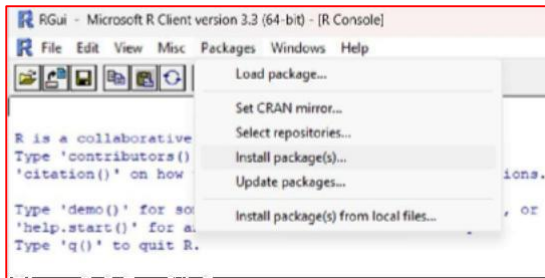
```
> #load data
```

```
> load("D:/titanic.raw.rdata")
```

```
> str(titanic.raw)
```

```
> load("D:/titanic.raw.rdata")
> str(titanic.raw)
'data.frame': 2201 obs. of 4 variables:
 $ Class : Factor w/ 4 levels "1st","2nd","3rd",...: 3 3 3 3 3 3 3 3 3 3 ...
 $ Sex   : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
 $ Age   : Factor w/ 2 levels "Adult","Child": 2 2 2 2 2 2 2 2 2 2 ...
 $ Survived: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Go to “**Packages**” select “**install packages**” select **India (Bengaluru)** [https] and select “**arules**”. Click on yes and then it will install the packages.



Code:

```
> library(arules)
```

```
> # find association rules with default settings
```

```
> rules <- apriori(titanic.raw)
```

```
> inspect(rules)
```

```
> library(arules)
Loading required package: Matrix

Attaching package: 'arules'

The following objects are masked from 'package:base':

  abbreviate, write
```

```
> rules <- apriori(titanic.raw)
Apriori

Parameter specification:
 confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext
 0.8      0.1    1 none FALSE          TRUE      5    0.1    1    10 rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
 0.1 TRUE TRUE FALSE TRUE    2    TRUE

Absolute minimum support count: 220

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].
sorting and recoding items ... [9 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [27 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
```

```
> inspect(rules)
```

	lhs	rhs	support	confidence	coverage	lift	count
[1]	{}	=> {Age=Adult}	0.9504771	0.9504771	1.0000000	1.0000000	2092
[2]	{Class=2nd}	=> {Age=Adult}	0.1185825	0.9157895	0.1294866	0.9635051	261
[3]	{Class=1st}	=> {Age=Adult}	0.1449341	0.9815385	0.1476602	1.0326798	319
[4]	{Sex=Female}	=> {Age=Adult}	0.1930940	0.9042553	0.2135393	0.9513700	425
[5]	{Class=3rd}	=> {Age=Adult}	0.2848705	0.8881020	0.3207633	0.9343750	627
[6]	{Survived=Yes}	=> {Age=Adult}	0.2971377	0.9198312	0.3230350	0.9677574	654
[7]	{Class=Crew}	=> {Sex=Male}	0.3916402	0.9740113	0.4020900	1.2384742	862
[8]	{Class=Crew}	=> {Age=Adult}	0.4020900	1.0000000	0.4020900	1.0521033	885
[9]	{Survived=No}	=> {Sex=Male}	0.6197183	0.9154362	0.6769650	1.1639949	1364
[10]	{Survived=No}	=> {Age=Adult}	0.6533394	0.9651007	0.6769650	1.0153856	1438
[11]	{Sex=Male}	=> {Age=Adult}	0.7573830	0.9630272	0.7864607	1.0132040	1667
[12]	{Sex=Female, Survived=Yes}	=> {Age=Adult}	0.1435711	0.9186047	0.1562926	0.9664669	316
[13]	{Class=3rd, Sex=Male}	=> {Survived=No}	0.1917310	0.8274510	0.2317129	1.2222950	422
[14]	{Class=3rd, Survived=No}	=> {Age=Adult}	0.2162653	0.9015152	0.2398910	0.9484870	476
[15]	{Class=3rd, Sex=Male}	=> {Age=Adult}	0.2099046	0.9058824	0.2317129	0.9530818	462
[16]	{Sex=Male, Survived=Yes}	=> {Age=Adult}	0.1535666	0.9209809	0.1667424	0.9689670	338
[17]	{Class=Crew, Survived=No}	=> {Sex=Male}	0.3044071	0.9955423	0.3057701	1.2384742	670
[18]	{Class=Crew, Survived=No}	=> {Age=Adult}	0.3057701	1.0000000	0.3057701	1.0521033	673
[19]	{Class=Crew, Sex=Male}	=> {Age=Adult}	0.3916402	1.0000000	0.3916402	1.0521033	862
[20]	{Class=Crew, Age=Adult}	=> {Sex=Male}	0.3916402	0.9740113	0.4020900	1.2384742	862
[21]	{Sex=Male, Survived=No}	=> {Age=Adult}	0.6038164	0.9743402	0.6197183	1.0251065	1329
[22]	{Age=Adult, Survived=No}	=> {Sex=Male}	0.6038164	0.9242003	0.6533394	1.1751385	1329
[23]	{Class=3rd, Sex=Male, Survived=No}	=> {Age=Adult}	0.1758292	0.9170616	0.1917310	0.9648435	387
[24]	{Class=3rd, Age=Adult, Survived=No}	=> {Sex=Male}	0.1758292	0.8130252	0.2162653	1.0337773	387
[25]	{Class=3rd, Sex=Male, Age=Adult}	=> {Survived=No}	0.1758292	0.8376623	0.2099046	1.2373791	387
[26]	{Class=Crew, Sex=Male, Survived=No}	=> {Age=Adult}	0.3044071	1.0000000	0.3044071	1.0521033	670
[27]	{Class=Crew, Age=Adult, Survived=No}	=> {Sex=Male}	0.3044071	0.9955423	0.3057701	1.2658514	670

Code:

```
> # rules with rhs containing "Survived" only
```

```
> rules <- apriori(titanic.raw, parameter = list(minlen=2, supp=0.005, conf=0.8),
  appearance = list(rhs=c("Survived=No", "Survived=Yes"), default="lhs"),
  control = list(verbose=F))
```

```
> rules.sorted <- sort(rules, by="lift")
```

```
> inspect(rules.sorted)
```

```
> rules.sorted <- apriori(titanic.raw, parameter = list(minlen=2, supp=0.005, conf=0.8), appearance = list(rhs=c("Survived=No", "Survived=Yes"), default="lhs"), control = list(verbose=F))
> rules.sorted <- sort(rules.sorted, by="lift")
> inspect(rules.sorted)
```

lhs	rhs	SUPPORT	CONFIDENCE	COVERAGES	LIFT	COUNT
(Class=2nd, Age=Child)	=> (Survived=Yes)	0.010904134	1.0000000	0.010904134	3.095660	24
(Class=2nd, Sex=Female, Age=Child)	=> (Survived=Yes)	0.003904604	1.0000000	0.003904604	3.095660	13
(Class=1st, Sex=Female)	=> (Survived=Yes)	0.046061790	0.9724130	0.045579146	3.010243	141
(Class=1st, Sex=Female, Age=Adult)	=> (Survived=Yes)	0.043607451	0.9722232	0.043424807	3.009450	140
(Class=2nd, Sex=Female)	=> (Survived=Yes)	0.042233521	0.8773555	0.040159927	2.715956	93
(Class=Crew, Sex=Female)	=> (Survived=Yes)	0.009006779	0.9495632	0.010445794	2.491261	20
(Class=Crew, Sex=Female, Age=Adult)	=> (Survived=Yes)	0.009006779	0.9495632	0.010445794	2.491261	20
(Class=2nd, Sex=Female, Age=Adult)	=> (Survived=Yes)	0.036347115	0.9402151	0.042233521	2.462516	80
(Class=2nd, Sex=Male, Age=Adult)	=> (Survived=No)	0.049948194	0.9166667	0.074328941	1.954053	154
(Class=2nd, Sex=Male)	=> (Survived=No)	0.049948194	0.9403352	0.051326670	1.270071	154
(Class=3rd, Sex=Male, Age=Adult)	=> (Survived=No)	0.175825149	0.8274623	0.209504590	1.237379	387
(Class=3rd, Sex=Male)	=> (Survived=No)	0.191701031	0.9274510	0.231712355	1.222295	422

Pruning Redundant Rules

In the above result, rule 2 provides no extra knowledge in addition to rule 1, since rule 1 tells us that all 2nd-class children survived. Generally speaking, when a rule (such as rule 2) is a super rule of another rule (such as rule 1) and the former has the same or a lower lift, the former rule (rule 2) is considered to be redundant. Below we prune redundant rules.

Code:

```
> # find redundant rules
```

```
> subset.matrix <- is.subset(rules.sorted, rules.sorted)
```

```
> subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
```

```
> redundant <- colSums(subset.matrix, na.rm=T) >= 1
```

```
> which(redundant)
```

```
> # remove redundant rules
```

```
> rules.pruned <- rules.sorted[!redundant]
```

```
> inspect(rules.pruned)
```

Output:

```

> robust.metrics[lower, c(robust.metrics, diag(4))] <- NA
Running message:
In '[c-1]'temp', as.vector(x), value = NA) :
N[.] <- TRUE: N is "bgMatrix", but not in (TRUE, FALSE) or constant NA (==> TRUE.
> rulescores <- mlRules(robust.metrics, na.rm=T) >= 1
> which(rulescores)
(Class*2nd, Age*Child, Survived*Yes) (Class*2nd, Sex*Female, Age*Child, Survived*Yes) (Class*1st, Sex*Female, Survived*Yes) (Class*1st, Sex*Female, Age*Adult, Survived*Yes)
(Class*2nd, Sex*Female, Survived*Yes) (Class*2nd, Sex*Female, Survived*Yes) (Class*2nd, Sex*Female, Age*Adult, Survived*Yes) (Class*2nd, Sex*Female, Age*Adult, Survived*Yes)
(Class*2nd, Sex*Male, Age*Adult, Survived*No) (Class*2nd, Sex*Male, Survived*No) (Class*3rd, Sex*Male, Age*Adult, Survived*No) (Class*3rd, Sex*Male, Survived*No)
> rules.growed <- rules.sorted(rulescores)
> names(rules.growed)
[1]

```