CLustering

Fiara Causo

11/3/2019

```
library(cluster)
library(tidyverse)
```

```
## -- Attaching packages -----------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts --------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(superheat)
# Read tab-delimited file by read.delim
car = read_delim("https://www.stat.berkeley.edu/~s133/data/cars.tab", delim = "\t")
```

```
## Parsed with column specification:
## cols(
##   Country = col_character(),
##   Car = col_character(),
##   MPG = col_double(),
##   Weight = col_double(),
##   Drive_Ratio = col_double(),
##   Horsepower = col_double(),
##   Displacement = col_double(),
##   Cylinders = col_double()
## )
```

```
# First 3 observations
head(car, 3)
```

```
## # A tibble: 3 x 8
##   Country Car      MPG Weight Drive_Ratio Horsepower Displacement Cylinders
##   <chr>   <chr>  <dbl>  <dbl>       <dbl>      <dbl>        <dbl>     <dbl>
## 1 U.S.    Buick~  16.9   4.36        2.73        155          350         8
## 2 U.S.    Ford ~  15.5   4.05        2.26        142          351         8
## 3 U.S.    Chevy~  19.2   3.60        2.56        125          267         8
```

the numeric variables need to be scaled scale() function helps with standardization. It has two more arguments other than passing in the data matrix. By center=TRUE, centering is done by subtracting the column means (omitting NAs); by scale=TRUE, s caling is done by dividing the (centered) columns by their standard deviations. If center=FALSE or scale=FALSE, no centering or scaling is done

```
# Standardize the variables by subtracting mean and divided by standard deviation
scar = scale(car[, -c(1,2)], center=TRUE, scale=TRUE)
```

```
set.seed(1)
#3-means clustering
```

```r
km = kmeans(scar, centers=3)
km
```

```
## K-means clustering with 3 clusters of sizes 8, 13, 17
##
## Cluster means:
##          MPG     Weight Drive_Ratio Horsepower Displacement  Cylinders
## 1 -1.1203872  1.4864539  -1.2960719  1.3712707   1.66759621  1.6252130
## 2 -0.5602631  0.2238870   0.2578965  0.3473723  -0.02489459  0.1376443
## 3  0.9556775 -0.8707154   0.4127012 -0.9109415  -0.76571412 -0.8700635
##
## Clustering vector:
##  [1] 1 1 1 1 3 3 3 3 2 2 2 2 2 2 2 2 1 1 1 1 3 2 3 3 3 3 3 3 2 2 3 3 3 3 3
## [36] 2 2 3
##
## Within cluster sum of squares by cluster:
## [1]  3.74590 24.08477 18.91804
##  (between_SS / total_SS =  78.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

K-Medoids Clustering based on centroids(medoids) bu calculating the absolute distance btw the points and the selected centroid rather than minimizing the squared euclidean distance

by using PAM the sums of the distances btw objects within a cluster are constantly recalculated as obvs move around unlike k-means that calculates the center of cluster after all obs have moved from one cluster to another

pam(): We can pass a data frame, or a distance matrix1 into the first argument x. The second argument k is the number of clusters that we wish to form. By keep.diss=TRUE, R is instructed to keep the dissimilarities. By keep.data=TRUE, the input data should be kept in the result. To compute a distance matrix, two functions can be used: dist() (in the default packages) and daisy (in cluster). dist() computes the distance matrix by using the specified distance measure. Commonly used distance measures are Euclidean distance (by specifying argument method="euclidean"), Manhattan distance (method="manhattan") and Maximum distance (method="maximum")

```r
# First, we use daisy to calculate the distance matrix
car.dist = daisy(scar)
print(car.dist)
```

```
## Dissimilarities :
##           1         2         3         4         5         6         7
## 2  1.1398481
## 3  1.8791302 1.5379233
## 4  0.8681787 0.6873475 1.5065957
## 5  6.5064614 6.4822961 5.1250923 6.3106342
## 6  5.1738042 5.0499221 3.8001685 4.9053397 1.8047804
## 7  5.5864282 5.6036243 4.3134390 5.4016818 1.2565441 1.0332439
## 8  6.1771844 6.0905876 4.7485982 5.9244671 0.7161097 1.2442626 1.0742030
## 9  4.8480549 5.0375972 3.8035553 4.8669738 2.3525992 2.1291365 1.6171913
## 10 3.5301708 3.7291650 2.6017142 3.5792512 3.5736168 2.6364429 2.6015184
## 11 4.9604936 5.1492658 3.9973838 4.9557693 2.3895043 1.8575477 1.3722800
```

```
## 12 3.3450719 3.6729516 2.6806678 3.5119282 3.9725777 3.0332883 2.9915492
## 13 3.0274764 2.7708193 1.5960990 2.7535396 3.7854036 2.3935780 3.0058421
## 14 3.9438666 3.7671366 2.4605709 3.7360814 2.8933537 1.9550300 2.3505016
## 15 2.7559271 2.5273572 1.4813731 2.5708289 4.1358672 2.7303071 3.3172245
## 16 2.4939094 2.2706162 1.3471376 2.2403426 4.3163472 2.8785776 3.4066196
## 17 1.4399982 0.8377919 0.7252240 1.0165631 5.7430227 4.3656972 4.9093636
## 18 1.7032564 0.9301185 0.7761175 1.1413136 5.7509851 4.3340633 4.9252674
## 19 1.2528877 0.2565576 1.3733210 0.6668156 6.3095525 4.8865115 5.4507553
## 20 1.2632414 0.7830240 0.8012940 0.7558885 5.7850435 4.4033377 4.9308012
## 21 5.2096828 5.0678428 3.8157918 4.9496450 1.6987044 0.3202520 1.0654982
## 22 3.7264974 3.6468502 2.2880581 3.5151842 2.9564835 1.7292372 2.0723052
## 23 6.9749041 6.9555364 5.5791028 6.7513604 0.7010289 2.2372431 1.7489593
## 24 6.4973291 6.3107820 4.9978582 6.1292098 1.7078072 1.6400328 1.8531241
## 25 5.4455210 5.2977556 4.0184339 5.1982029 1.5478613 0.6090464 1.2158603
## 26 6.7281549 6.7264044 5.3582423 6.5269933 0.3924042 2.0229042 1.3853449
## 27 6.2516001 6.0358394 4.7099391 5.9344674 1.2582953 1.2889114 1.5271840
## 28 5.1216822 4.8108545 3.6618306 4.7373244 2.5974309 1.0604331 2.0770315
## 29 4.1658959 3.9606554 2.6594600 3.7477335 3.1072917 1.6871993 2.3063279
## 30 3.9568990 3.8096329 2.4692785 3.6037691 3.0034600 1.5925941 2.1310964
## 31 5.4980541 5.2816705 4.0886676 5.1187130 2.3371058 1.1814421 1.9882024
## 32 6.4810802 6.3980437 5.0519523 6.2111056 0.9131638 1.6397765 1.5291032
## 33 6.8079280 6.7733290 5.3989540 6.5929239 0.3825189 2.0451145 1.5351385
## 34 6.7428379 6.5979100 5.2646847 6.4146012 1.6110434 1.9546073 2.0929158
## 35 6.3291275 6.3396799 4.9879301 6.1412506 0.3891058 1.6270124 0.9753294
## 36 4.5310683 4.6441960 3.2882995 4.4744300 2.3271473 1.9982563 1.7047030
## 37 5.0541206 5.1541528 3.9678998 4.9833487 2.1643824 1.5764399 1.1034142
## 38 6.7948022 6.7891309 5.4196563 6.5851179 0.4869201 2.0802109 1.4464682
##            8         9        10        11        12        13        14
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  2.4465576
## 10 3.4329055 1.5014408
## 11 2.3609068 0.8441935 1.7107844
## 12 3.8451138 1.8036762 0.5255723 1.9532352
## 13 3.3916875 2.7160211 1.9461582 2.8241709 2.2036527
## 14 2.6358656 2.0212696 1.8617590 2.3614848 2.2640202 1.1593258
## 15 3.7620304 2.8598682 1.9126425 2.9541083 2.0712991 0.5003866 1.4839209
## 16 3.9437224 2.9561575 1.8881155 3.0078993 2.0407394 0.7494156 1.7424394
## 17 5.3618900 4.3540948 3.0918327 4.5183655 3.1008994 2.0619163 3.0077541
## 18 5.3391575 4.4700081 3.2336889 4.6116307 3.2820741 2.0714550 3.0411856
## 19 5.9121360 4.9282952 3.6531977 5.0486036 3.6284414 2.6105509 3.6010959
## 20 5.4002110 4.3981637 3.1371642 4.5417264 3.1329324 2.1526641 3.1127122
## 21 1.1892967 2.0608821 2.6384138 1.8743475 3.0478077 2.3563277 1.8038583
## 22 2.6248563 1.7958162 1.3032347 1.9312784 1.7386606 1.1905000 1.0047733
## 23 1.0204637 2.9535998 4.1460710 2.9540731 4.5385930 4.2941996 3.4462460
## 24 1.1176406 3.3610043 4.0842400 3.1635894 4.5058524 3.7556124 3.2240898
## 25 1.0272409 2.2123917 2.8808062 2.0843180 3.2789801 2.5547655 1.9284228
## 26 0.8979276 2.5433909 3.7767096 2.5459045 4.1703806 4.0638586 3.1984995
## 27 0.7226258 2.8135472 3.6275852 2.7364046 4.0695291 3.3295052 2.6087595
```

```
## 28 1.9380432 3.0442380 3.2111523 2.8027787 3.5918097 2.2947691 2.1662867
## 29 2.5543986 2.8421867 2.5390312 2.7460931 2.9094252 1.8375723 1.9726533
## 30 2.5124425 2.4557730 2.0997795 2.4032013 2.4798109 1.5608200 1.6455885
## 31 1.6387728 3.2186993 3.5788929 2.9627948 3.9517206 2.8169783 2.5990762
## 32 0.5399900 2.8951122 3.8782418 2.8172944 4.2791128 3.7347060 3.0063712
## 33 0.8409380 2.6961640 3.9043616 2.7177677 4.3053914 4.0843817 3.2041305
## 34 1.1505897 3.5150291 4.3530385 3.3837812 4.7533608 4.0043668 3.3981805
## 35 0.6590117 2.1966443 3.3872917 2.1515380 3.7731194 3.6747621 2.8546416
## 36 2.3239170 0.8284025 1.4496585 1.4566540 1.8538570 2.2807905 1.4742083
## 37 2.0955544 0.9293048 1.7974353 0.4185086 2.1255921 2.7296457 2.2054957
## 38 0.9445111 2.6284194 3.8529893 2.6245026 4.2496167 4.1342707 3.2746391
##           15        16        17        18        19        20        21
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16 0.6135479
## 17 1.8523603 1.6612209
## 18 1.9038060 1.7164735 0.3484249
## 19 2.4022245 2.1432108 0.6896808 0.7442321
## 20 1.9721830 1.7262461 0.3112976 0.4990653 0.6197894
## 21 2.6970359 2.8720900 4.3735675 4.3486218 4.9026316 4.4256685
## 22 1.4643705 1.5574226 2.9028656 2.9276306 3.4979870 2.9585980 1.7360496
## 23 4.6691439 4.8427560 6.2148382 6.2075757 6.7729493 6.2406187 2.1896065
## 24 4.1601229 4.3029176 5.6084492 5.5235895 6.1172285 5.6294160 1.7273423
## 25 2.8872833 3.1291034 4.5853313 4.5589804 5.1302201 4.6490493 0.4130040
## 26 4.4209623 4.5764749 5.9890670 5.9944815 6.5523856 6.0187706 1.9654050
## 27 3.7063800 3.9155152 5.3046264 5.2447759 5.8512434 5.3707284 1.1890624
## 28 2.6437355 2.8307240 4.1643391 4.0630542 4.6330340 4.2186003 1.1173955
## 29 2.2156802 2.2263245 3.2752130 3.1804593 3.7742653 3.2678478 1.8515108
## 30 1.9244810 1.9416802 3.1008883 3.0456623 3.6340760 3.1046275 1.7263843
## 31 3.2043755 3.3602005 4.6324672 4.5398596 5.0925051 4.6450790 1.3165811
## 32 4.1237846 4.3128218 5.6710966 5.6389351 6.2104759 5.6973056 1.6161605
## 33 4.4473685 4.6289818 6.0286118 6.0253378 6.5951862 6.0669668 1.9667330
## 34 4.4047432 4.6039058 5.8810526 5.8127673 6.4018476 5.9033347 1.9892055
## 35 4.0185214 4.1758140 5.6107231 5.6215377 6.1714477 5.6390014 1.5802116
## 36 2.5011695 2.6304055 3.9000247 3.9936022 4.5083666 3.9492086 1.9208685
## 37 2.9037674 2.9578242 4.5033203 4.5699679 5.0407289 4.5386834 1.5785176
## 38 4.4968791 4.6449792 6.0525231 6.0544862 6.6134046 6.0798614 2.0305467
##           22        23        24        25        26        27        28
## 2
## 3
## 4
## 5
```

```
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17
## 18
## 19
## 20
## 21
## 22
## 23 3.4695843
## 24 3.0862555 1.5893300
## 25 1.9739830 2.0167725 1.6235892
## 26 3.1851298 0.4691402 1.6994356 1.8416265
## 27 2.6810030 1.5170768 1.0286972 0.9756777 1.4653244
## 28 2.0864283 2.9200943 1.8463675 1.1870636 2.8254805 1.6302850
## 29 1.3888145 3.3986058 2.4827899 2.0676480 3.2487747 2.5191228 1.6192914
## 30 0.9567097 3.3675915 2.6313055 1.9686886 3.1685641 2.5300071 1.7122281
## 31 2.4659351 2.4796851 1.2935247 1.3109424 2.4841532 1.5030967 0.8534042
## 32 3.0224100 0.8164795 0.9631897 1.4177507 0.9588893 0.9580309 2.1679101
## 33 3.2507493 0.3618590 1.6148982 1.7937362 0.2851090 1.3269554 2.7692642
## 34 3.3773124 1.3394565 0.6691116 1.7771885 1.5986781 1.1984472 2.1748994
## 35 2.7966626 0.8094290 1.6246464 1.4827076 0.4532477 1.3225642 2.4994953
## 36 1.3182650 2.8822778 3.1423812 2.0508011 2.5349410 2.6201402 2.7813383
## 37 1.8002414 2.7506750 2.8780007 1.8108202 2.3356329 2.4112481 2.5306662
## 38 3.2520390 0.4240946 1.6773707 1.9123832 0.1103997 1.4947954 2.8688498
##           29        30        31        32        33        34        35
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17
## 18
## 19
## 20
## 21
```

```
## 22
## 23
## 24
## 25
## 26
## 27
## 28
## 29
## 30 0.4464760
## 31 1.7406712 1.9194350
## 32 2.7848997 2.8042141 1.6824079
## 33 3.2848474 3.2218676 2.4278263 0.8318586
## 34 2.8599053 2.9846217 1.5536115 0.7295724 1.4493989
## 35 2.9065222 2.8009512 2.2045161 0.9086064 0.5976211 1.5967788
## 36 2.3532768 1.9696566 2.9421027 2.7289716 2.6452250 3.2869352 2.2009025
## 37 2.5584726 2.2356866 2.7234010 2.5836214 2.4942355 3.1418917 1.9433425
## 38 3.2824562 3.2131639 2.5086130 0.9674219 0.3093247 1.5799187 0.5369531
##           36        37
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17
## 18
## 19
## 20
## 21
## 22
## 23
## 24
## 25
## 26
## 27
## 28
## 29
## 30
## 31
## 32
## 33
## 34
## 35
## 36
## 37 1.4103762
```

```
## 38 2.6134362 2.4109537
##
## Metric :  euclidean
## Number of objects : 38
```

```
# can also do: car.dist = dist(scar)
# 3-medoids clustering
scar.pam = pam(scar, k=3, keep.diss=TRUE)
scar.pam
```

```
## Medoids:
##      ID        MPG      Weight Drive_Ratio Horsepower Displacement
## [1,] 20 -1.0020180  1.36815073 -1.24294794  1.2578275   1.58320969
## [2,]  8  0.9377088 -0.89534762  0.53428969 -1.0110385  -0.81336768
## [3,] 22 -0.4369007  0.06663918 -0.02592652  0.2746522  -0.07076625
##       Cylinders
## [1,]  1.6252130
## [2,] -0.8700635
## [3,]  0.3775747
## Clustering vector:
##  [1] 1 1 1 1 2 2 2 2 3 3 3 3 3 3 3 3 1 1 1 1 2 3 2 2 2 2 2 2 3 3 2 2 2 2 2
## [36] 3 3 2
## Objective function:
##    build     swap
## 1.137333 1.031719
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

Hierarchical Clustering -pre specify the number of clusters K -creats a tree based observation called a dendrogram

hclust() by passing the distance matrix into it. By default, R uses complete linkage (defaulting method="complete"). We can use single linkage (by specifying method="single") and average linkage (method="average") too. The cluster library provides a similar function, called agnes to perform hierarchical cluster analysis. Naturally, the first step is calculating a distance matrix. Again, there are two functions that can be used to calculate distance matrices, dist() and daisy(). The Hierarchical clustering is shown below.

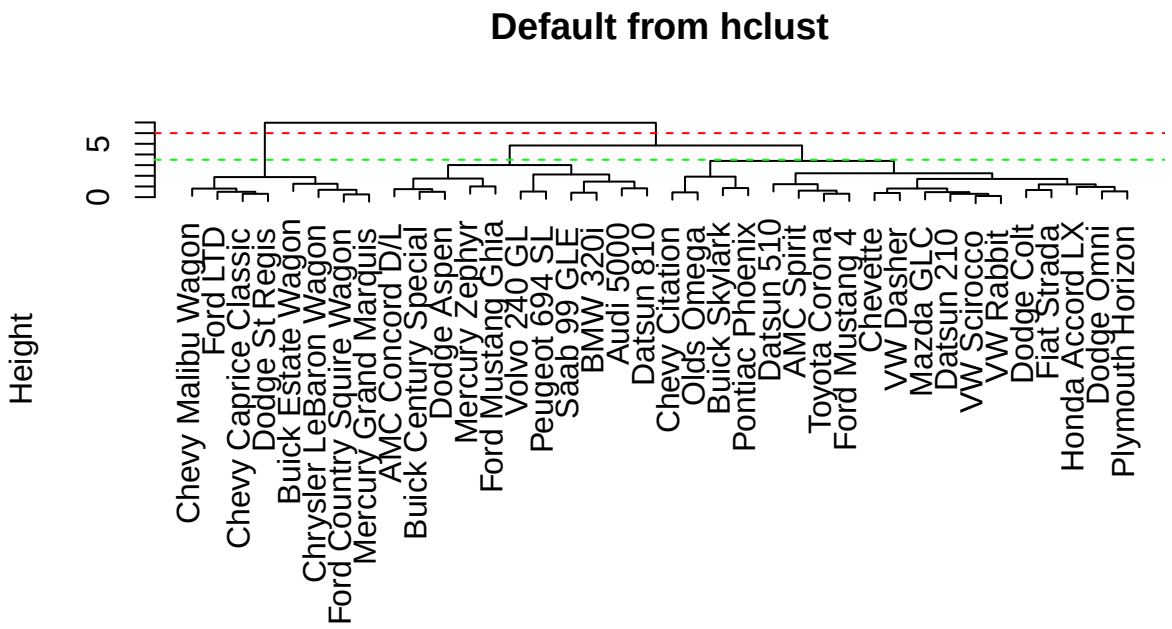
```
# We use the dist function in this example.
car.dist = dist(scar)
# Can also do: car.dist = daisy(scar)
# Agglomerative Hierarchical clutering
set.seed(1)
car.hclust = hclust(car.dist) # complete linkage
# Can also do: car.hclust = agnes(car.dist)
```

Complete distance it means that when the cluster is formed, its distance to other objects is computed as the max distance btw any obect in the cluster and the other object

This is a tree-like display that lists the objects which are clustered along the x-axis, and the distance at which the cluster was formed along the y-axis. (Distances along the x-axis are meaningless in a dendrogram; the observations are equally spaced to make the dendrogram easier to read.)

```
# Plot dendogram
plot(car.hclust, labels=car$Car, main='Default from hclust')
# Add a horizontal line at a certain height
```

```
abline(h=6, col="red", lty=2)
abline(h=3.5, col="green", lty=2)
```




```
clus = cutree(car.hclust, 3)
table(clus)
```

```
## clus
##  1  2  3
##  8 19 11
```

We can use table to compare the results of the hclust() and pam() solutions:

```
table(hclust=clus, pam=scar.pam$clustering)
```

```
##       pam
## hclust  1  2  3
##      1  8  0  0
##      2  0 17  2
##      3  0  0 11
```

```
car$Car[clus != scar.pam$clustering]
```

```
## [1] "Chevy Citation" "Olds Omega"
```