# Content-Based Recommendation with R

EarthPleabian

In this project, we will understand how a content-based recommendation system works and a simple example will be conducted.In this project, we will understand how a content-based recommendation system works through a simple example. The dataset that will be used is the wine dataset from https://archive.ics.uci.edu/ml/datasets/wine.

This dataset is the result of the chemical analysis of wine grown in the same region in Italy. We have data from three different cultivars (From an assemblage of plants selected for desirable characters).

To extract the data from UCI machine learning repository we can use:

```
library(data.table)
wine.data <-fread('https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data')
head(wine.data)
```

```
##    V1    V2   V3   V4   V5  V6   V7   V8   V9  V10  V11  V12  V13  V14
## 1:  1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29 5.64 1.04 3.92 1065
## 2:  1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
## 3:  1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
## 4:  1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
## 5:  1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.04 2.93  735
## 6:  1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.05 2.85 1450
```

We have a total of 14 columns. The column number 1, named V1 represents the cultivar. The distribution of the V1 column:

```
table(wine.data$V1)
```

```
##
##  1  2  3
## 59 71 48
```

Next we separate between cultivar (`wine.type`) and the chemical properties of the wine (`wine.features`). `wine.features` has all the properties and without the cultivar column. Let's scale this `wine.features` and create a matrix

```
wine.type <- wine.data[,1]
wine.features <- wine.data[,-1]

wine.features.scaled <- data.frame(scale(wine.features))
wine.mat <- data.matrix(wine.features.scaled)
```

We have converted our data frame to a matrix. Next we add the row names and give an integer number for each wine:

```
rownames(wine.mat) <- seq(1:dim(wine.features.scaled)[1])
wine.mat[1:2,]
```

```
##          V2         V3         V4        V5         V6        V7        V8
## 1 1.5143408 -0.5606682  0.2313998 -1.166303 1.90852151 0.8067217 1.0319081
```

```
## 2 0.2455968 -0.4980086 -0.8256672 -2.483841 0.01809398 0.5670481 0.7315653
##           V9        V10        V11       V12      V13       V14
## 1 -0.6577078  1.2214385  0.2510088 0.3611585 1.842721 1.0101594
## 2 -0.8184106 -0.5431887 -0.2924962 0.4049085 1.110317 0.9625263
```

We want to find the pearson coefficient between the rows. We want the similarity between two wines. Hence we will transpose our matrix before invoking `cor` function.

```
wine.mat <- t(wine.mat)
cor.matrix <- cor(wine.mat, use = "pairwise.complete.obs", method ="pearson")
dim(cor.matrix)
```

```
## [1] 178 178
```

```
cor.matrix[1:5,1:5]
```

```
##           1          2         3             4             5
## 1 1.0000000  0.7494842 0.5066551  0.7244043066  0.1850897291
## 2 0.7494842  1.0000000 0.4041662  0.6896539740 -0.1066822182
## 3 0.5066551  0.4041662 1.0000000  0.5985843958  0.1520360593
## 4 0.7244043  0.6896540 0.5985844  1.0000000000 -0.0003942683
## 5 0.1850897 -0.1066822 0.1520361 -0.0003942683  1.0000000000
```

Here, our output is the similarity between the different wines. The `cor.matrix` matrix is the similarity matrix, which shows how closely related items are. The values range from -1 for perfect negative correlation, when two items have attributes that move in opposite directions, and +1 for perfect positive correlation, when attributes for the two items move in the same direction. For example, in row 1, wine 1 is more similar to wine 2 than wine 3. The diagonal values will be +1, as we are comparing a wine to itself.

Next, let's do a recommendation test:

```
user.view <- wine.features.scaled[3,]
user.view
```

```
##          V2         V3       V4         V5         V6        V7       V8
## 3 0.1963252 0.02117152 1.106214 -0.2679823 0.08810981 0.8067217 1.212114
##          V9      V10       V11       V12       V13      V14
## 3 -0.497005 2.129959 0.2682629 0.3174085 0.7863692 1.391224
```

Let's a say a particular user is either tasting or looking at the properties of wine 3. We want to recommend him wines similar to wine 3.

```
sim.items <- cor.matrix[3,]
sim.items
```

```
##           1           2           3           4           5           6
##  0.50665507  0.40416617  1.00000000  0.59858440  0.15203606  0.54025182
##           7           8           9          10          11          12
##  0.57579895  0.18210803  0.42398729  0.55472235  0.66895949  0.40555308
##          13          14          15          16          17          18
##  0.61365843  0.57899194  0.73254986  0.36166695  0.44423273  0.28583467
##          19          20          21          22          23          24
##  0.49034236  0.44071794  0.37793495  0.45685238  0.48065399  0.52503055
##          25          26          27          28          29          30
##  0.41103595  0.04497370  0.56095748  0.38265553  0.36399501  0.53896624
##          31          32          33          34          35          36
##  0.70081585  0.61082768  0.37118102 -0.08388356  0.41537403  0.57819928
##          37          38          39          40          41          42
##  0.33457904  0.50516170  0.34839907  0.34398394  0.52878458  0.17497055
```

```
##          43          44          45          46          47          48
##  0.63598084  0.10647749  0.54740222 -0.02744663  0.48876356  0.59627672
##          49          50          51          52          53          54
##  0.68698418  0.48261764  0.76062564  0.77192733  0.50767052  0.41555689
##          55          56          57          58          59          60
##  0.40473005  0.70494822  0.44715598  0.63943883  0.69278870 -0.24448643
##          61          62          63          64          65          66
## -0.51953343 -0.69852162 -0.39656304  0.37786280 -0.34110515  0.51114086
##          67          68          69          70          71          72
##  0.40081393 -0.07133864 -0.51892414  0.17118942 -0.17150288  0.16010450
##          73          74          75          76          77          78
## -0.38354357  0.05870863  0.46296109 -0.25389622  0.03181391 -0.53159057
##          79          80          81          82          83          84
##  0.29123936 -0.16622364  0.02006811  0.45673413 -0.07130762 -0.51844220
##          85          86          87          88          89          90
##  0.74758861  0.08404697 -0.23871593 -0.16710688 -0.14204149 -0.04964413
##          91          92          93          94          95          96
## -0.13449668 -0.24586805 -0.31281621  0.27647644  0.20556767  0.29470573
##          97          98          99         100         101         102
##  0.05446738  0.30165673  0.49909615  0.27479163  0.02617475 -0.09327926
##         103         104         105         106         107         108
## -0.07085997 -0.13272233  0.04588610 -0.46453748  0.09638495 -0.29022380
##         109         110         111         112         113         114
##  0.15251643  0.69015248  0.44738637 -0.10868289 -0.31883798  0.02548044
##         115         116         117         118         119         120
## -0.18138987 -0.02370276  0.06135685 -0.19550928 -0.54299911  0.04608302
##         121         122         123         124         125         126
##  0.39368261  0.08087597 -0.27454459 -0.02623270  0.46757807 -0.04204135
##         127         128         129         130         131         132
##  0.16195497 -0.23107700 -0.01632784 -0.23324605 -0.20122178 -0.44997667
##         133         134         135         136         137         138
## -0.47035013 -0.37045822 -0.36482041 -0.60645494 -0.50822763 -0.61677576
##         139         140         141         142         143         144
## -0.72800937 -0.74334463 -0.60964408 -0.55125765 -0.71566410 -0.63573916
##         145         146         147         148         149         150
## -0.29424284 -0.47005923 -0.55427831 -0.51609209 -0.46264959 -0.44707308
##         151         152         153         154         155         156
## -0.31048758 -0.24936792 -0.19692793 -0.32385020 -0.42974804 -0.49009374
##         157         158         159         160         161         162
## -0.39696754 -0.52698001 -0.06039994 -0.06567643 -0.49971124 -0.73796076
##         163         164         165         166         167         168
## -0.74001697 -0.55986619 -0.53577637 -0.65819119 -0.47990747 -0.45215617
##         169         170         171         172         173         174
## -0.35228074 -0.37906137 -0.63437150 -0.56822118 -0.38398329 -0.53273823
##         175         176         177         178
## -0.49107312 -0.43566842 -0.41977671 -0.52506466
```

We look at the third row in our similarity matrix. We know that the similarity matrix has stored all the item similarities. So the third row gives us the similarity score between wine 3 and all the other wines. The preceding results are truncated.

We want to find the closest match:

```
sim.items.sorted <- sort(sim.items, decreasing = TRUE)
sim.items.sorted[1:5]
```

```
##         3        52        51        85        15
## 1.0000000 0.7719273 0.7606256 0.7475886 0.7325499
```

First, we sort row 3 in decreasing order, so we have all the items close to wine 3 popping to the front. Then we pull out the top five matches. Great–we want to recommend wines 52, 51, 85, and 15 to this user. We ignore the first recommendation as it will be the same item we are searching for. In this case, the first element will be wine 3 with a similarity score of 1.0.

Let's look at the properties of wine 3 and the top five matches to confirm our recommendation:

```
rbind(wine.data[3,], wine.data[52,], wine.data[51,], wine.data[85,], wine.data[15,])
```

```
##    V1    V2   V3   V4   V5  V6   V7   V8   V9  V10  V11  V12  V13  V14
## 1:  1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
## 2:  1 13.83 1.65 2.60 17.2  94 2.45 2.99 0.22 2.29 5.60 1.24 3.37 1265
## 3:  1 13.05 1.73 2.04 12.4  92 2.72 3.27 0.17 2.91 7.20 1.12 2.91 1150
## 4:  2 11.84 0.89 2.58 18.0  94 2.20 2.21 0.22 2.35 3.05 0.79 3.08  520
## 5:  1 14.38 1.87 2.38 12.0 102 3.30 3.64 0.29 2.96 7.50 1.20 3.00 1547
```

you can see that the wine properties in our recommendation are close to the properties of wine 3.