

7.7: Answers to exercises

Answer to the stars question.

First, load the data and as suggested above, convert coordinates into decimals:

Next, some preliminary plots (please make them yourself):

Now, load [dbscan](#) package and try to find where number of “constellations” is maximal:

Plot the prettified “night sky” (Figure 7.7.1) with found constellations:

```
dev.off()
```

To access agreement between two classifications (two systems of constellations) we might use adjusted Rand index which counts correspondences:

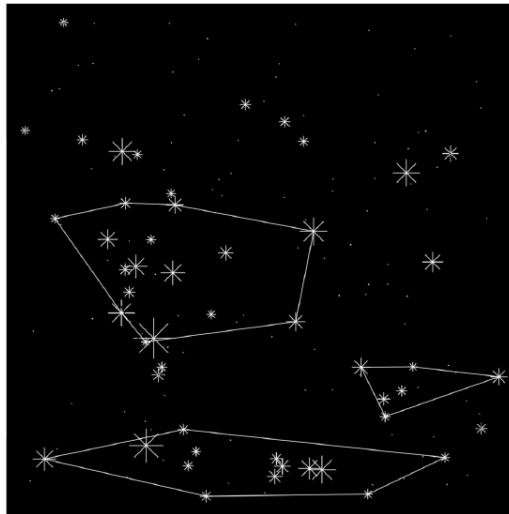


Figure 7.7.1 Fifty brightest stars with “constellations” found with DBSCAN.

(It is of course, low.)

Answer to the beer classification exercise. To make hierarchical classification, we need first to make the distance matrix. Let us look on the data:

Data is binary and therefore we need the specific method of distance calculation. We will use here Jaccard distance implemented in [vegdist\(\)](#) function from the [vegan](#) package. It is also possible to use here other methods like “binary” from the core [dist\(\)](#) function.

Next step would be the construction of dendrogram (Figure 7.7.2):

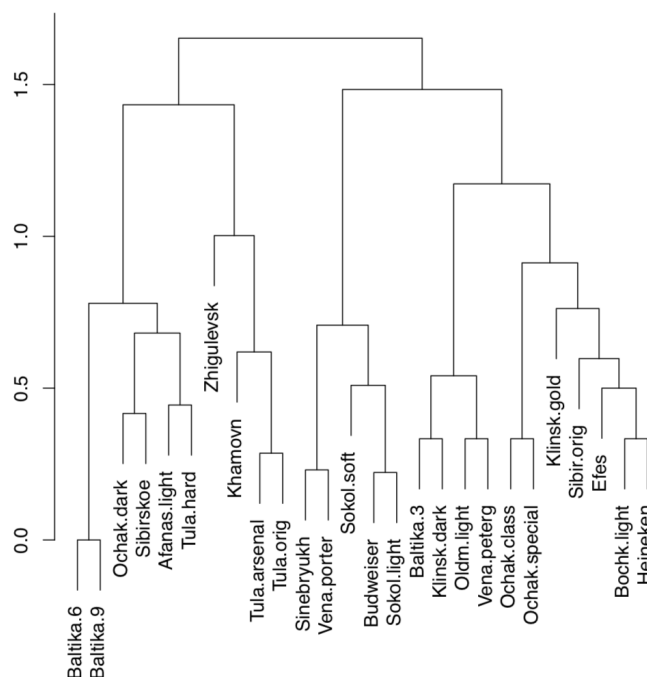


Figure 7.7.2 Hierarchical classification of Russian beer types.

There are two big groups (on about 1.7 dissimilarity level), we can call them “Baltika” and “Budweiser”. On the next split (approximately on 1.4 dissimilarity), there are two subgroups in each group. All other splits are significantly deeper. Therefore, it is possible to make the following hierarchical classification:

- Baltika group
 - Baltika subgroup: Baltika.6, Baltika.9, Ochak.dark, Afanas.light, Sibirskoe, Tula.hard
 - Tula subgroup: Zhigulevsk, Khamovn, Tula.arsenal, Tula.orig
- Budweiser group
 - Budweiser subgroup: Sinebryukh, Vena.porter, Sokol.soft, Budweiser, Sokol.light
 - Ochak subgroup: Baltika.3, Klinsk.dark, Oldm.light, Vena.peterg, Ochak.class, Ochak.special, Klinsk.gold, Sibir.orig, Efes, Bochk.light, Heineken

It is also a good idea to check the resulted classification with any other classification method, like non-hierarchical clustering, multidimensional scaling or even PCA. The more consistent is the above classification with this second approach, the better.

Answer to the plant species classification tree exercise. The tree is self-explanatory but we need to build it first (Figure 7.7.3):

Answer to the kubricks (Figure 7.7.3) question. This is just a plan as you will still need to perform these steps individually:

1. Open R, open Excel or any spreadsheet software and create the data file. This data file should be the table where kubrick species are rows and characters are columns (variables). Every row should start with a name of kubrick (i.e., letter), and every column should have a header (name of character). For characters, short uppercased names with no spaces are preferable.

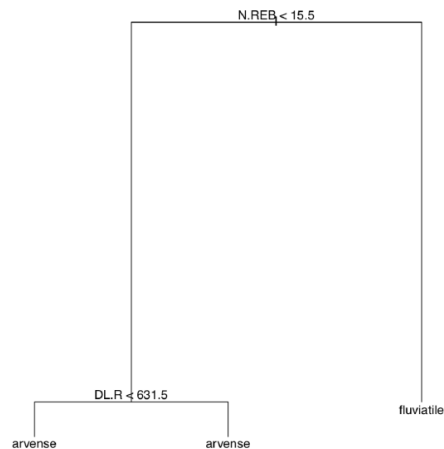


Figure 7.7.3 Classification tree shows that two horsetail species differ by **N.REB** character (number of stem ridges)

Topleft cell might stay empty. In every other cell, there should be either 1 (character present) or 0 (character absent). For the character, you might use “presence of stalk” or “presence of three mouths”, or “ability to make photosynthesis”, or something alike. Since there are 8 kubricks, it is recommended to invent $N + 1$ (in this case, 9) characters.

2. Save your table as a text file, preferably tab-separated (use approaches described in the second chapter), then load it into R with `read.table(..., h=TRUE, row.names=1)`.

3. Apply hierarchical clustering with the distance method applicable for binary (0/1) data, for example binary from `dist()` or another method (like Jaccard) from the `vegan::vegdist()`.

4. Make the dendrogram with `hclust()` using the appropriate clustering algorithm.

In the data directory, there is a data file, `kubricks.txt`. It is just an example so it is not necessarily correct and does not contain descriptions of characters. Since it is pretty obvious how to perform hierarchical clustering (see the “beer” example above), we present below two other possibilities.

First, we use MDS plus the MST, *minimum spanning tree*, the set of lines which show the shortest path connecting all objects on the ordination plot (Figure 7.7.4):

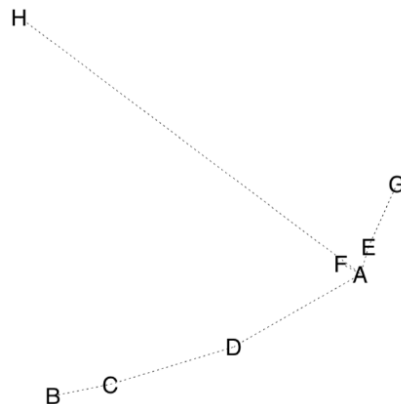


Figure 7.7.4 Minimum spanning tree of kubricks.

Second, we can take into account that kubricks are biological objects. Therefore, with the help of packages `ape` and `phangorn` we can try to construct the most parsimonious (i.e., shortest) *phylogeny tree* for kubricks. Let us accept that kubrick H is the *outgroup*, the most primitive one:

(Make and review this plot yourself.)

7.7: Answers to exercises is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.