Università Ca'Foscari Venezia

Corso di Laurea magistrale (*ordinamento ex D.M. 270/2004*)

in Informatica – Computer Science

Tesi di Laurea

—

Ca' Foscari
Dorsoduro 3246
30123 Venezia

Statistical methods for Lead Optimization in drug discovery problems

Relatrice
Ch.ssa Prof.ssa Irene Poli

Laureando
Marco Cattapan
Matricola 987154

Anno Accademico
2013 / 2014

INDEX

CHAPTER 1
Presentation of the problem

1.1 The lead optimization phase in drug discovery

New drug discovery is usually a long and complex process because it involves several phases and the presence of a high number of relevant variables including categorical ones can make difficult the phases of analysis, modelling and experimentation on the data.

A key phase of this process concerns the generation of modulators of protein function that are small molecules that binding to a protein, alter and modulate the activity because when proteins are linked with other molecules, their conformation can change slightly or relevantly. [1] [2] [3]

This phase is done under the hypothesis that this activity can affect a particular disease state.

Current practices rely on the screening of vast libraries of small molecules (often 1-2 million molecules) in order to identify a molecule which specifically inhibits or activates the protein function, commonly known as a Lead Molecule (LM) and it serves as a starting point for developing a new drug.

The lead molecule generally lacks the other attributes needed for a drug candidate, such as: absorption, distribution, metabolism and excretion (ADME). [1]

These attributes enable the molecule to be dosed orally, reach the required site of action in the body and stay around long enough to have an effect.

In addition, the molecule should be safe avoiding activities that cause adverse effects. . [4]

In order to achieve these attributes, retaining the interaction capacity with the target protein, the lead molecule must be modified through the phase called Lead optimization.

This optimization is accomplished through chemical modification of the molecule structure (hit structure), by involving long synthesis and testing cycles with

modifications chosen by employing knowledge of the structure-activity relationship (SAR) as well as quantitative structure activity relationships (QSAR). [1] [4] [5]
The structure-activity relationship (SAR) is the relationship between the chemical or 3D structure and its biological activity and using these information allows to determine the chemical groups for achieving a certain effect on a biological target organism and then to modify the effect of a drug by varying its chemical structure. [6]
This quantitative structure activity relationships (QSAR) is a mathematical relation that quantitatively expresses the biological activity of a drug as a function of certain physical-chemical or structural features of the molecule that are used as predictor variables.
In addition, models based on QSA allow to predict the biological activity of new drugs. [7]
This is a multi-objective optimization and complex high dimensional problem in pharmaceutical research: the chemical space in which teams operate is very large, the objectives tend to compete (for example, solubility and absorption are inversely correlated) and some of the criteria (for example, metabolism) are both expensive to measure and difficult to predict.
The traditional approach consists in an long iterative procedure among formulation, synthesis, and testing cycles involving high investment of resources in terms of experimental units and time to reach the target that can lead to a possible negative impact on the environment.
In the last few years have been proposed computational approaches for the SAR and QSAR analyses mostly based upon machine learning techniques.
In addition, evolutionary algorithms (genetic algorithms, evolutionary programming, evolution strategies) for searching and optimization have been presented and applied with success in the drug discovery process.
The objective of lead optimization is to reach the optimal value of the variable of interest conducting a very small number of tests reducing thus the investments on resources. [1]

Focusing on the phase of lead optimization, the themes of classification and clustering are analyzed for the construction of models for the identification of the most useful variables for this purpose.

1.2 Description of data

The thesis is focused mainly on the topics of classification and clustering on data provided to the research center European Centre for Living Technology (ECLT) in Venice from a well known pharmaceutical company with headquarters in London as subject of a European research project (FET area: future emerging technologies).

The project involves the analysis of molecules for the formulation of a new drug against childhood leukemia.

The company has provided to the research center 2 dataset for this project.

The first dataset is a library of 2500 molecules identified by two reagents (*Atag* and *Btag* components) that define their chemical composition on which 5 measurements have been calculated for each molecule.

Focusing on this dataset, it is composed by:

- 2 independent categorical variables characterized by 50 levels each: *Atag* and *Btag* represent specific different components of a molecule
- 5 dependent variables: *pIC50 MMP12 - Activity, gskmw, clogP*, *ADMET Solubility* and *TOPKAT Rat Oral LD50*

Then each data point, representing a molecule, is described by two categorical variables defined as reagents, each of which can assume 50 different modalities.

The whole experimental space is therefore identified by 2500 experimental points (50^2) without replications in the experiments.

In our analysis, we will give more importance to the most important response variable that is *pIC50 MMP12* which measures the molecular activity of the reaction product.

Its numerical values are recorded for only 1704 experiments while the remaining experiments assume different labels: "Assay Failed" (28), "Inactive" (176), "Not Assayed" (26) and "Not Mate" (566).

We denoted as "*Activity"* the variable *pIC50 MMP12* with only numerical values to make clear this distinction.

The variable *Activity* assumes values between 3.7 and 8 and we are primarily interested in molecules associated to high values for this variable because the intent of the analysis is to find the reaction whose product maximises the molecular activity.

On the other hand, the values of the other 4 response variables are reported for all the 2500 experimental points.

The second dataset provided by the pharmaceutical company contains a similarity matrix among molecules (example "A01B04" against "A01B06").

The matrix size is 2500x2500 and it shows how one experiment is chemically similar to all the other experiments in terms of Tanimoto distance.

What we are interested in is to understand if molecules closer in the chemical composition (which means molecules with higher values in the similarity matrix), have similar *Activity* values with, as consequence, low distance between their responses.

Having to deal with also categorical variables which makes the problem rather complicated, the analysis will be quite difficult and for this reason will be used also some statistical methods that can handle this kind of variables.

1.3 Exploratory data analysis

Before starting with the analysis, we have to extract the main features from these response variables to describe these data.

We have conducted a preliminary descriptive analysis aiming to identify specific patterns among the chemical compounds.

The information extracted by this analysis will be then used to understand which methodology will more effective and efficient in modeling the data.

The simple descriptive statistics of the responses are summarized in *Table 1* and the same information can be extracted looking their boxplots in *Figure 1.*

A boxplot is a graphical representation used to describe the distribution of data through simple indices of dispersion and position.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max	sd
Activity-pIC50 MMP12	3.700	4.600	5.300	5.461	6.300	8.000	1.025
gskmw	291.3	416.4	448.5	454.1	483.5	728.9	57.427
clogP	-2.505	2.656	3.731	3.635	4.609	9.142	1.503
ADMET Solubility	-8.280	-5.353	-4.554	-4.628	-3.817	-1.766	1.117
TOPKAT Rat Oral LD50	0.618	2.005	2.316	2.280	2.571	3.626	0.445

Table 1: simple descriptive statistics

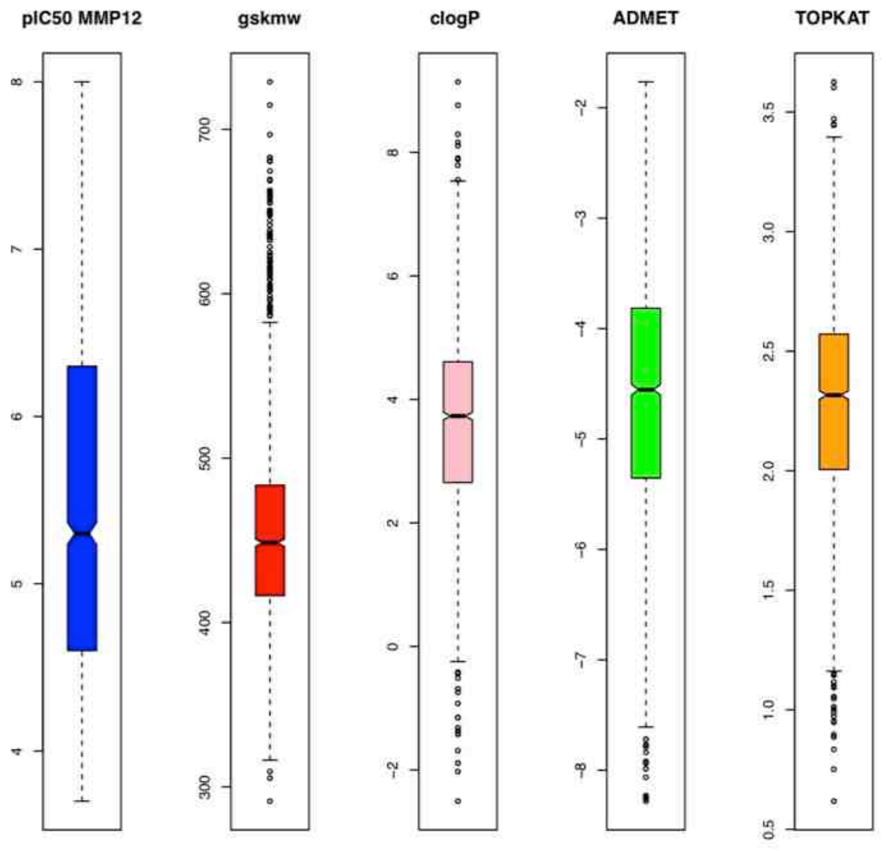


Figure 1: boxplots of response variables

A measure of variability based on percentiles is given by the interquartile difference Q_3-Q_1 that is just the width of the box.

Since between Q_1 and Q_3 there is the median value that is the central point of the distribution, if their difference is small this means that the variability is limited, otherwise is high. [8]

At this point, we can say that the response variable *pIC50 MMP12* has the maximum variability while the other variables seems to have a quite symmetric distribution.

Except for the first variable, there are some outlier values, in particular in the variable *gskmw*.
An outlier is an anomalous value that unbalances the distribution and it lies more than one and a half time the length of the box from either extremities, so above a threshold defined by $Q_1 - 1.5*(Q_3 - Q_1)$ or $Q_3 + 1.5*(Q_3 - Q_1)$.

The smoothed estimated density distributions of response variables are plotted in *Figure 2* to see how their values are distributed.
Focusing on our main variable of interest *pIC50 MMP12-Activity*, we can note that we have the highest density of values in the range between 4.5 and 5.5 approximately.
The shape of its density function confirms the absence of a symmetrical distribution.
Its global density distribution highlights a bimodal behavior of the variable's frequencies if we consider two groups of molecules: red line identifies the distribution of the variable for the molecules with high *Activity* (≥ 6.8) while the blue line for molecules with low *Activity* (< 6.8).
Looking at the other variables, mostly *gskmw* presents a quite long tail for the presence of outliers seen in the previous boxplot that makes the distribution less symmetric.

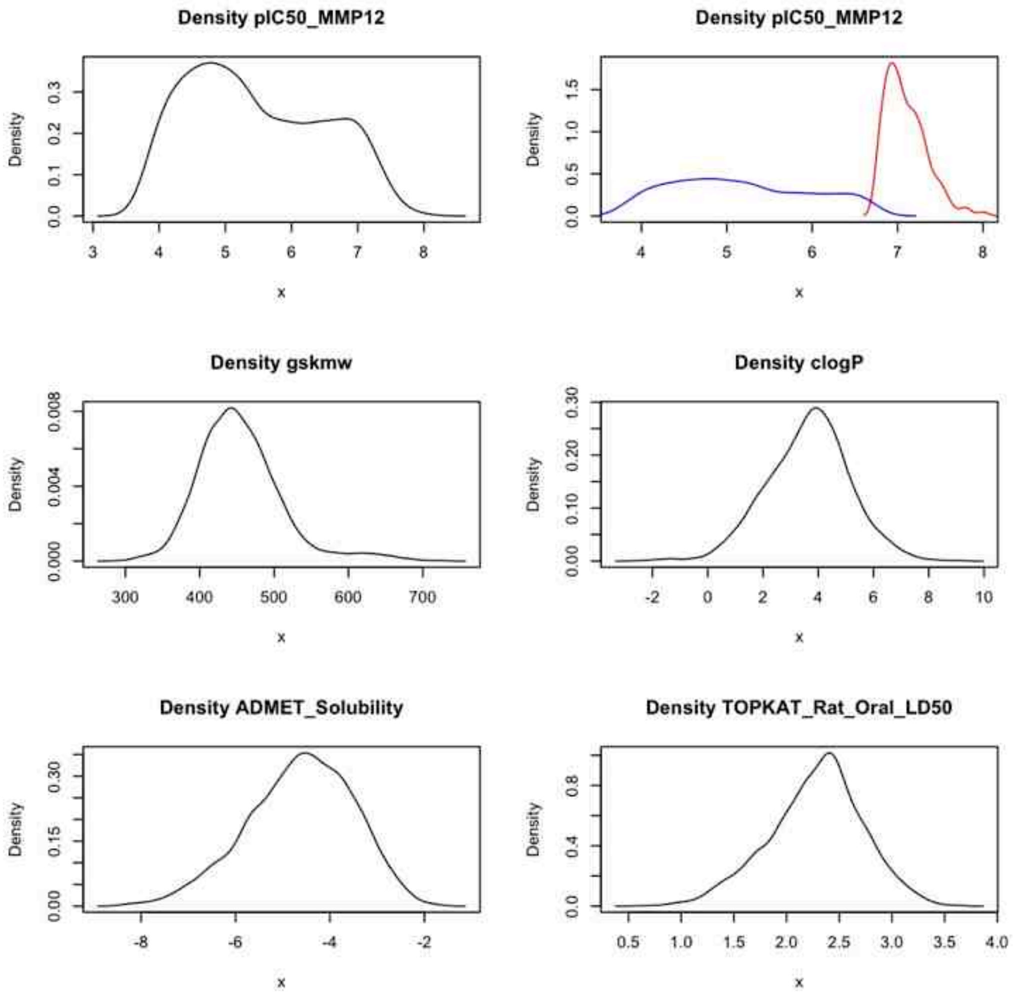


Figure 2: smoothed estimated density distributions

A correlation analysis among the response variables has been performed to identify if and which dependent variables can be expressed by linear relationship with another one.

The linear correlation expresses the intensity of the bond between the two variables and it can measured by the Pearson coefficient.

This coefficient always takes values between -1 and 1 and if it is greater than 0, this means that the two variables are positively/directly correlated otherwise they are negatively/inversely correlated.

Positively correlated means that if a variable grows, then the other variable grows as well, while in the other case we have that the other variable decreases.

Table 2 show that some response variables are particularly related, in particular:

- *ADMET Solubility* vs *gskmw* = -0.518 (moderate correlation)
- *ADMET Solubility* vs *clogP* = -0.806 (strong correlation)

	pIC50 MMP12	gskmw	clogP	ADMET	TOPKAT
pIC50 MMP12	1	-0.058	0.040	0.068	-0.130
gskmw	-0.058	1	0.459	-0.518	-0.156
clogP	0.040	0.459	1	-0.806	0.262
ADMET	0.069	-0.518	-0.806	1	-0.339
TOPKAT	-0.130	-0.156	0.262	-0.339	1

Table2: correlation analysis

Another way to obtain these conclusions from the correlation matrix is to use the scatter plot matrix which aim is to investigate particular behaviors considering the variables in pair, as we can see in *Figure 3*.

A scatter plot pairs up the values of the two variables in the same observation as (x,y) coordinates in a Cartesian diagram.

If the points are arranged as a cloud, this means that there isn't a particular relationship between the variables.

In the other case, the points are approximately arranged as a diagonal and its direction gives us the information the type of correlation, so positive or negative.

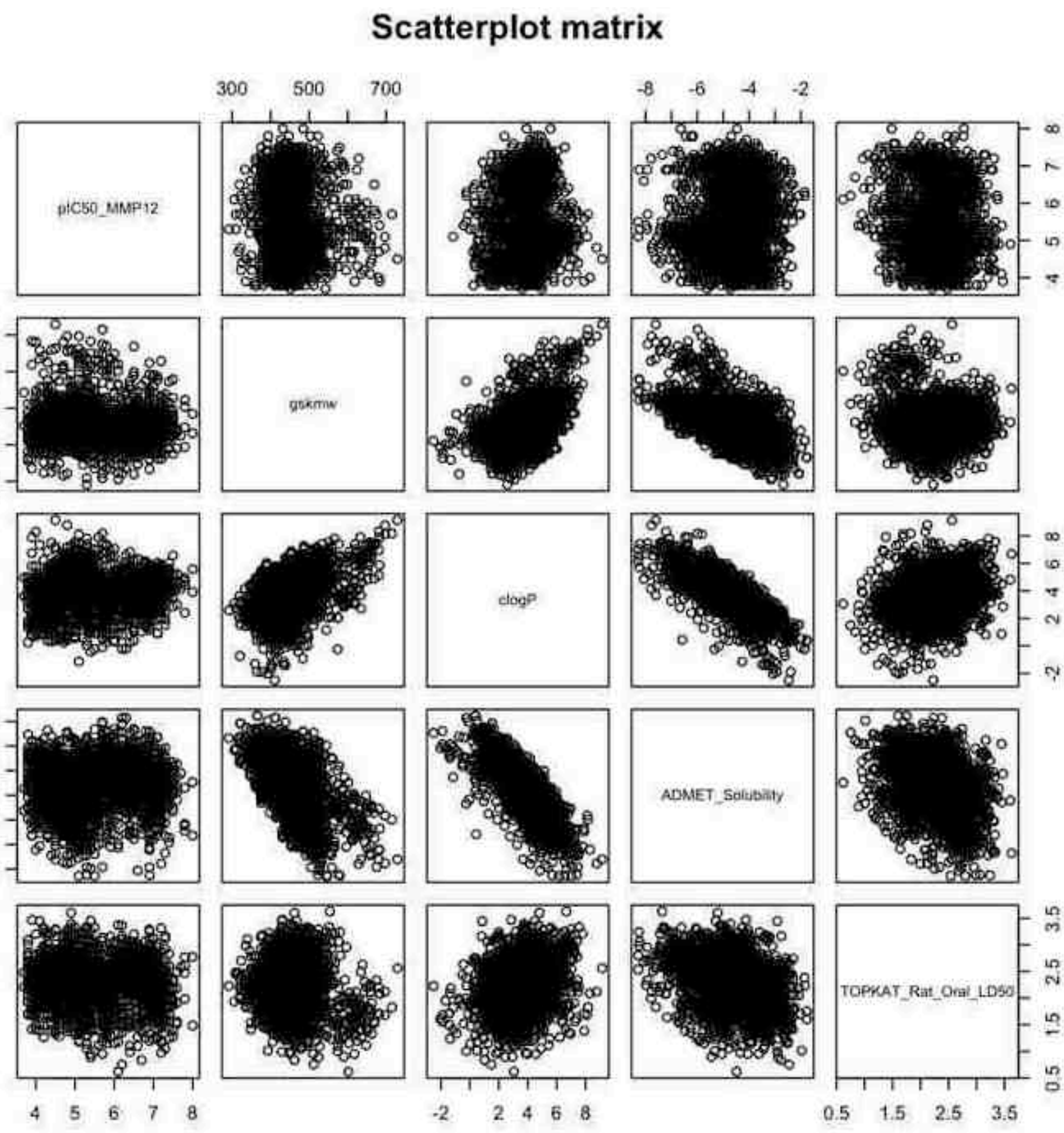


Figure 3: scatter plot matrix

We further investigate the behavior of *Activity* variable (numeric values from *pIC50 MMP12*) trying to understand if a specific molecule (which means a specific combination of *Atag* and *Btag* levels) is influenced by the presence of a particular *Atag* or *Btag* modality.

We used the paired boxplots to observe if there are differences in the values distribution of *Activity* variable, taking a fixed component (*Atag* or *Btag* variable).

For a correct analysis, we have also to consider for each level of the fixed variable, the number of modalities of the other variable in which is present a value.

The behavior of *Activity* fixing each level of *Atag* is reported in *Figure 4(a)* where the number above each boxplot indicates the modalities of *Btag* associated to that specific *Atag* level.
For example, the first boxplot in the figure represent the distribution of 39 molecules composed by the element A01.
We note that there is no significant difference in position of the boxplots.

Instead, looking at the behavior of the boxplots fixing each level of *Btag*, there is a large variability in their positions.
Then, we put the boxplots in ascending order according to their median value to easily see the trend, as illustrated in *Figure 4(b)*.
For reasons of layout of this figure, we report directly in the table the *Btag* levels in ascending order with respect to the median value with the number of *Atag* modalities associated to that specific *Btag* level.
The boxplots with the highest median values of the distribution correspond to B18, B25 and B19 which are levels of *Btag* with many modalities of *Atag* associated (45, 42 and 43 respectively) and we have to take them into account.
After this analysis on these boxplots, we can conclude that the *Btag* component seems to highly effect the Activity values of the molecules.

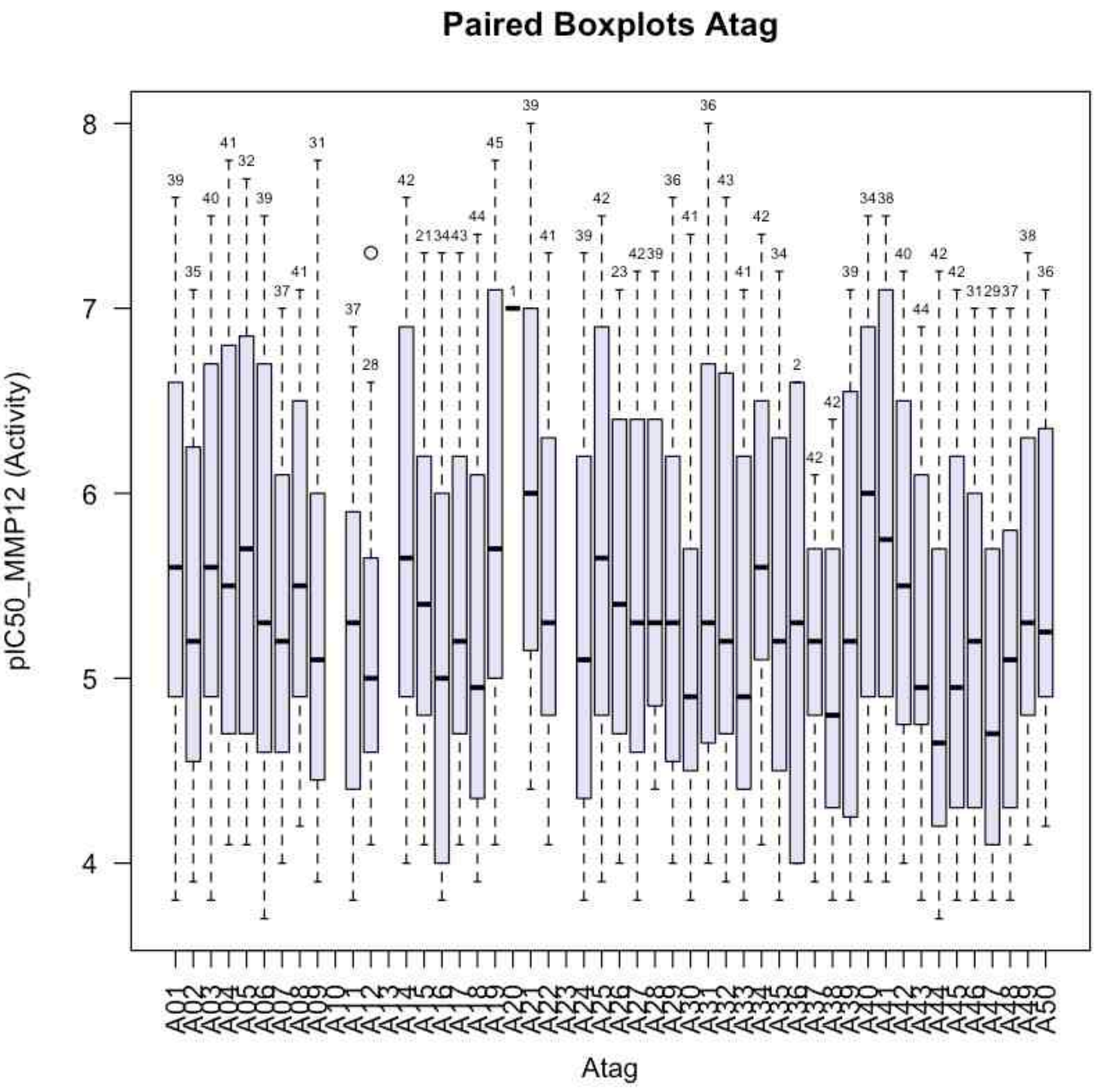
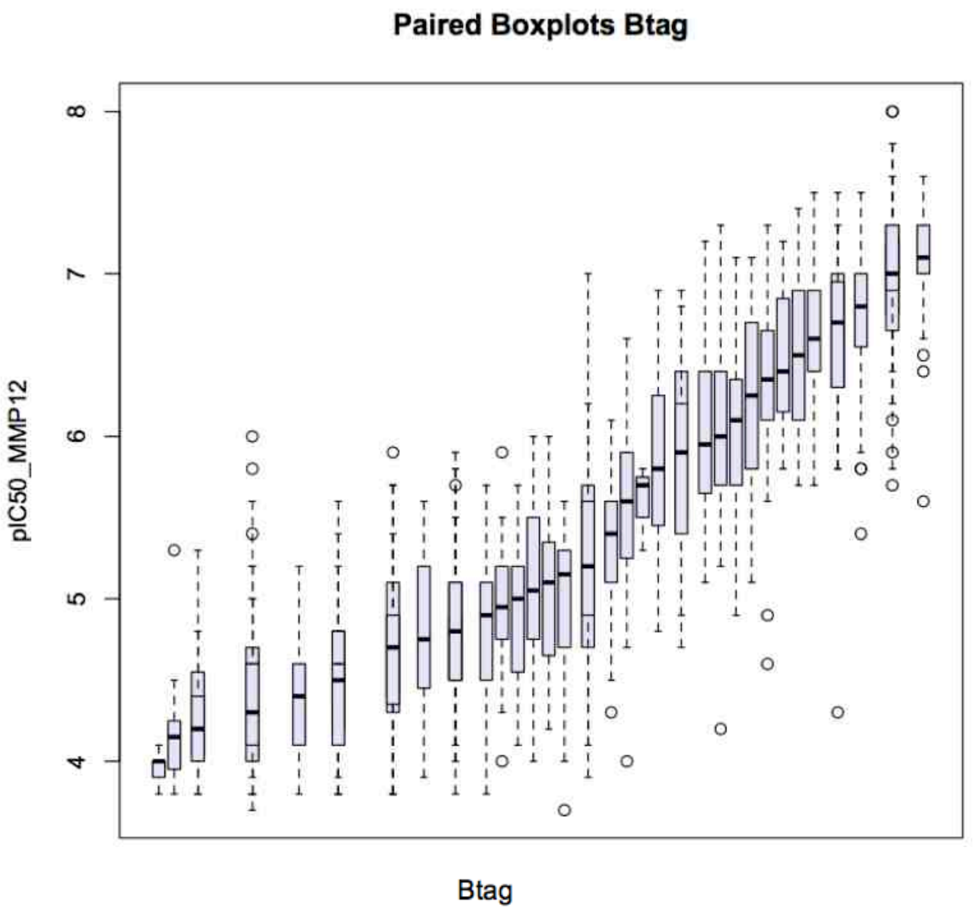


Figure 4(a): behavior of Activity fixing each level of Atag



B26	B39	B22	B32	B09	B16	B36	B37	B49	B13	B28	B34	B46	B50	B05	B14	B21	B24	B29	B44	B47	B42	B35	B08	B38
9	20	27	22	21	35	14	26	35	33	31	31	37	6	39	37	40	36	37	41	41	42	40	39	36
B40	B06	B03	B31	B10	B15	B45	B11	B12	B43	B01	B30	B23	B17	B41	B33	B27	B20	B04	B48	B02	B07	B19	B25	B18
19	34	42	44	41	44	3	39	41	42	44	14	43	38	36	36	37	41	42	43	40	36	43	42	45

Figure 4(b): behavior of Activity fixing each level of Btag

It is possible to describe the main response variable *Activity* with a 3D-plot that can help to identify regions of the experimental space where the highest response is more likely to be found. [9]

As the *Activity* depends on the combination of the two independent variables *Atag* and *Btag*, a heat map representing the *Activity* values for each combination is derived in *Figure 5*.

The variable *Btag* is reported on the x-axis whereas the variable *Atag* is on the y-axis.

The point in which one *Atag* level intersects with one *Btag* level represents the activity of the resulting molecule.

Higher values of *Activity* are represented by blue squares (pixels) and smaller values by light blue squares as reported in the legend.

White pixels correspond to combinations of components in which the response variable *pIC50 MMP12* has as value a string.
The heat map emphasizes that the *Btag* variable has a more homogeneous behavior because the response *Activity* is more similar if we consider the columns.
Immediately, we see that only very few molecules have a high *Activity* value.
In particular, only two optima experiments (A21B07 and A31B25) achieve the maximum value (8) of the response variable (circled in red).
Looking at the columns associated to *Btag*, the molecules between B17 and B20 (red rectangle) reach very good responses but they not include the two global maxima.

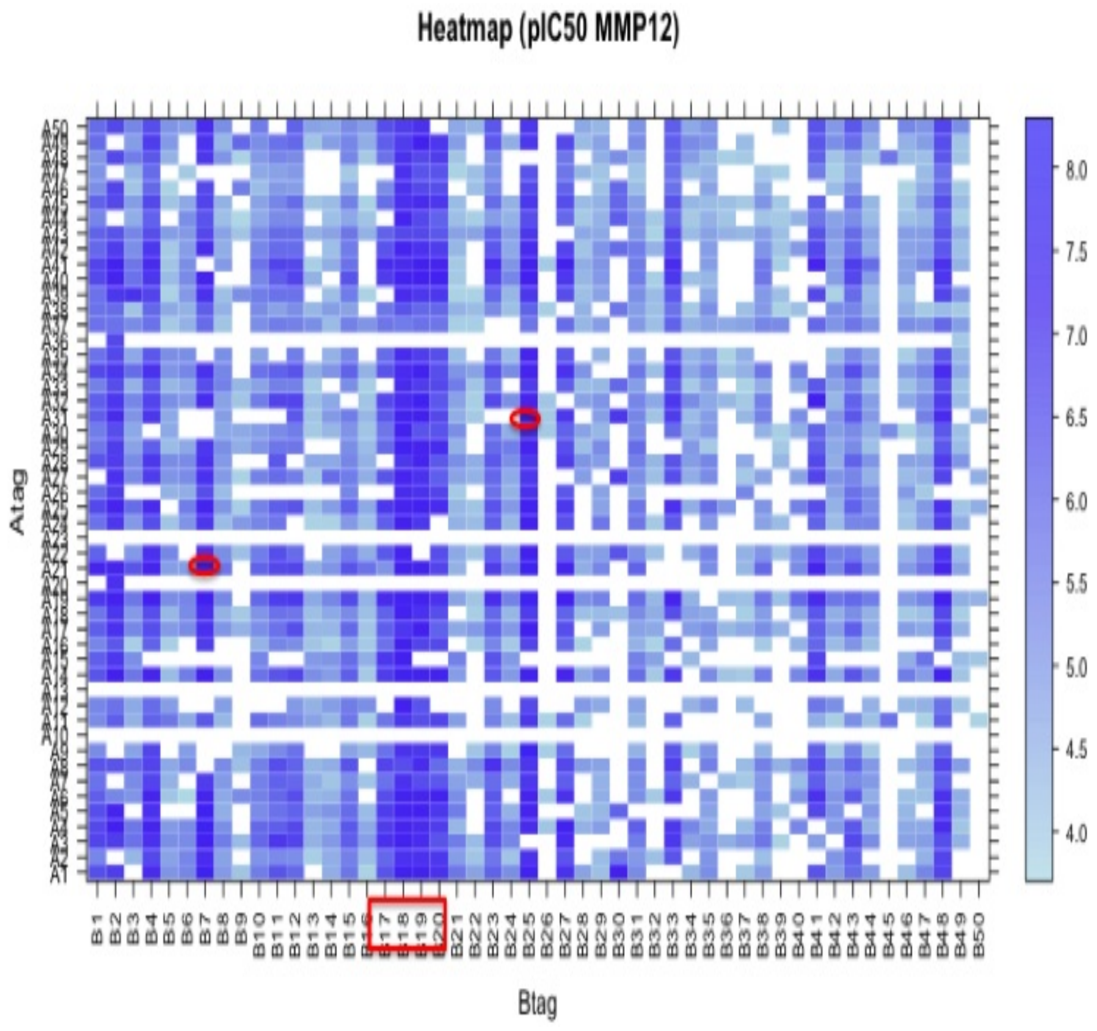


Figure 5: heat map of Activity

As said before, the pharmaceutical company provided us a second dataset, collecting a similarity matrix among molecules based on Tanimoto distance.

The values express how one experiment is chemically similar to all the other experiments.

As done for the previous dataset, we compute a preliminary descriptive analysis on the similarity matrix considering all the 2500 molecules, so we take into account molecules that don't have a numeric value on the variable *pIC50 MMP12* but rather a label.

The symmetric similarity matrix has obviously the main diagonal of ones.

The descriptive statistics on the similarity matrix reported in *Table 2* don't consider the main diagonal in order to not distort the values.

Min	1st Qu.	Median	Mean	3rd Qu.	Max	sd
0.1429	0.2564	0.2887	0.3020	0.3271	0.8667	0.075

Table 2: simple descriptive statistics on similarity matrix

Looking at these indexes, mean and median are pretty close to each other and this suggests that the distribution of the similarity values is quite symmetric.

Moreover, the values related to the quantiles are rather low and this means that the molecules are not so close in chemical composition.

These aspects can be seen also through the density distribution of the similarity matrix in which gives us an idea of how its values are distributed, plotted in *Figure 6*.

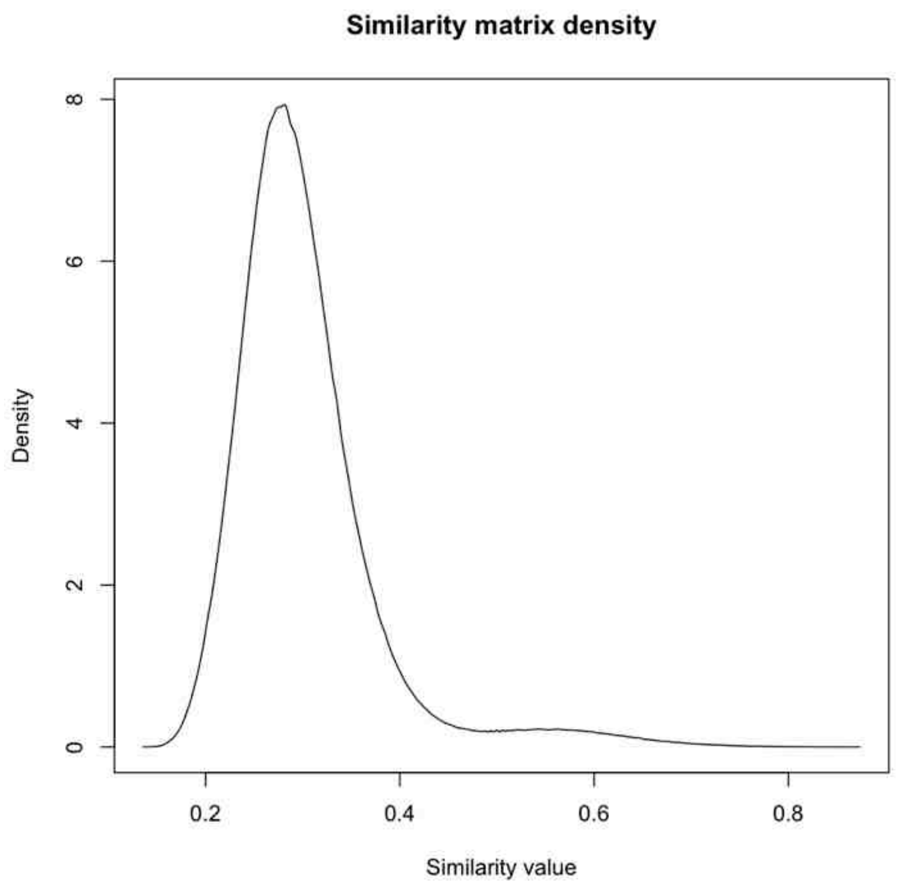


Figure 6: density distribution of similarity values

The highest density is in the range between 0.2 and 0.4 approximately (91.16% of similarity values).

As consequence, we will have to do mainly with low similarity values thus with molecules not so much similar to each other chemically.

Using the heap map representing the similarity values for each combination of molecules, we can take note some clusters but for the large amount of data is difficult to extrapolate other information.

After analyzing the two matrices, we have to study how and how much the Activity variable changes, fixing the *Atag* component as varying the *Btag* component and viceversa.

We have created a new matrix collecting the absolute differences in the variable *Activity* among experiments.

In the following graphs, the similarity values between molecules are reported on x-axis whereas the differences of *Activity* are positioned on y-axis.

In *Figure 7* we examine this relation, considering for example the two molecules where we have the maximum value in *Activity* variable (A21B27) and its minimum value (A06B06), respectively.
For each graph, the refence molecule is put in relation with at most 50 experiments that have in common one of the two components *Atag* or *Btag* and we plot their response distances against the similarity values.
The graphs on the left represent the comparisons keeping fixed the *Atag* component of the specified molecule but *Btag* varies (for example in the graph on the top left, the molecule A21B07 is compared with molecules in which *Atag* variable is A21 and *Btag* component ranges from 01 to 50).
Instead on the other side, the comparisons are made keeping fixed the *Btag* component and varying *Atag* (for example in the graph on the top right, the molecule A21B07 is compared with molecules in which *Atag* ranges from 01 and 50 and *Btag* is B07).
An important aspect to note making these types of comparisons is that we find practically all the highest similarity values related to the molecule used as reference.
In fact, these similarity values are so high that they correspond to the right tail of the similarity distribution seen previously.
This confirms that we have high similarity values when these molecules are formed by one of the components of the reference molecule, in other words if they are chemically similar.

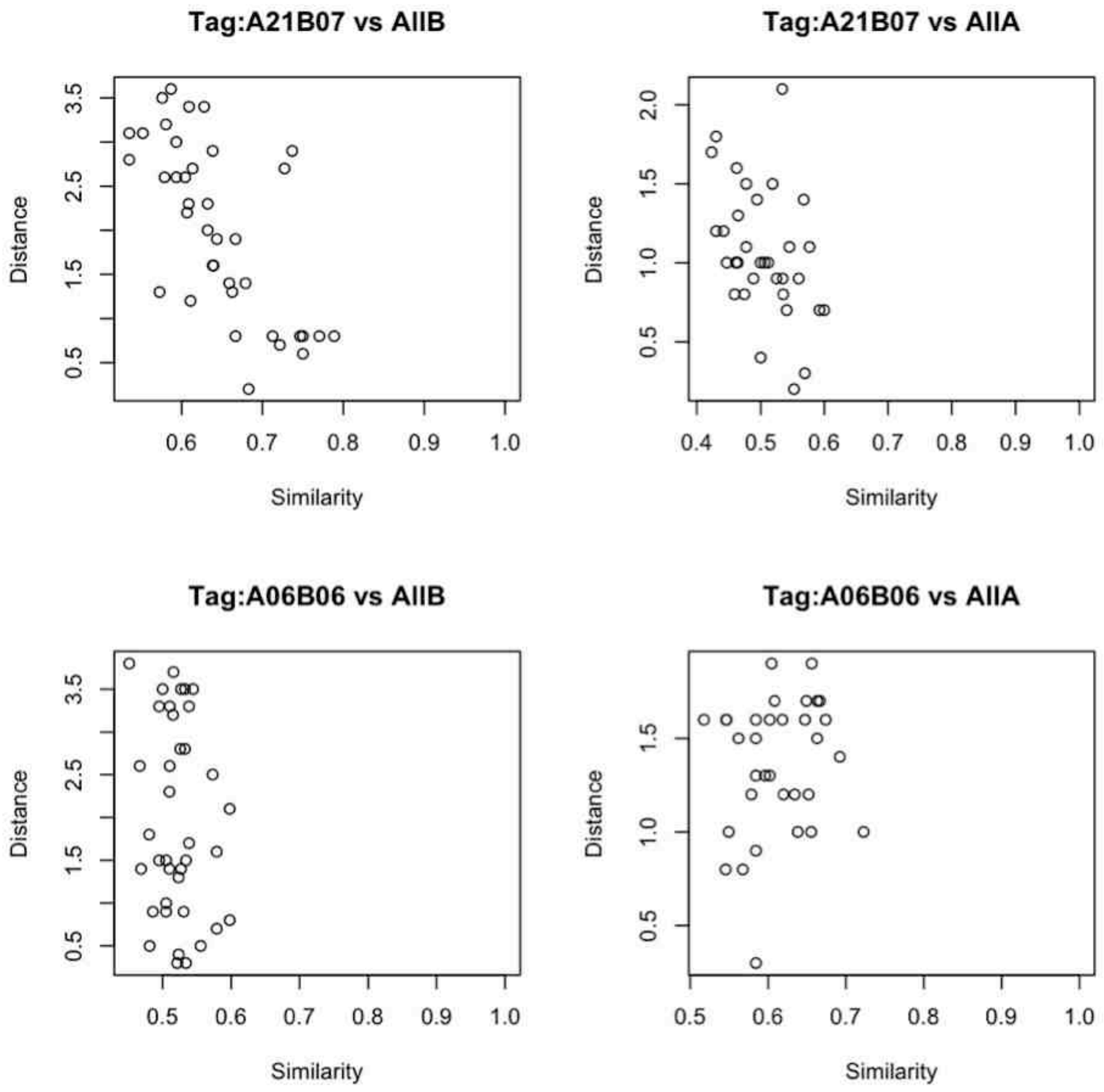


Figure 7: relation between similarity and distance values

As we said before, it is interesting to understand if molecules closer in the chemical composition have low distance between their *Activity* values.

Looking at the plots, especially where *Btag* varies, we can intuit a decreasing trend which means that if the similarity increases, then the distance in the responses tends to decrease.

Going further in details, we can consider 100 randomly molecules to have an idea if there is this global decreasing behavior, as shown in the *Figure 8*.

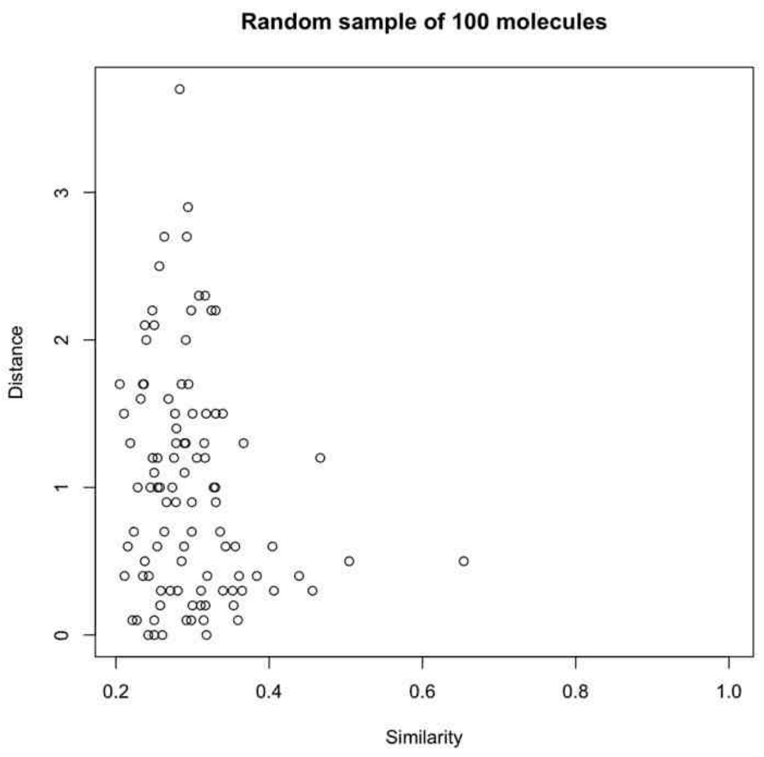


Figure 8: relation considering 100 random molecules

It is difficult to realize this behavior using simple plots, we well see if this intuition is confirmed in the chapter about the clustering.

1.4 Some highlights from exploratory data analysis

After this preliminary descriptive analysis, the most interesting conclusions that we can drawn are:

- levels between B17 and B20 are characterized by high *Activity* values
- *Btag* component highly effects the *Activity* behavior
- Large amount of low similarity values between the range [0.2,0.4] and this aspect will be considered in the following chapters
- High similar values when the molecules have one component in common
- Some elements show a possible relationship between similarity and distance values among molecules: if the similarity increase, the distance in the response variable tends to decrease

CHAPTER 2
Classification approach

The theme of classification has been dealt with in this thesis for the identification of classes of molecules with specific behaviors and these classes produced by the algorithm will be compared with predefined classes from the pharmaceutical company.

2.1 Supervised learning: classification

In statistical methods exists a main distinction between supervised and unsupervised learning algorithms.

Classification is an instance of supervised learning where the training set consists of a set of labelled training examples in which each observation is composed by its features and a label representing its class.

A supervised learning algorithm analyzes the training data and produces a model to assign a class to the unlabelled observations belonging to the test set.

The objective of the learning phase is to understand this mapping between data and mapping

Then, the aim of the supervised classification is to search a decision criterion for the classification of new units based on their characteristics.

These new observations are assigned to g groups/classes (known in advance) according to the values of p explanatory variables $x_1, \dots, x_p$ measured on them.

In this context, we have to find methods able to associate new molecules to predefined classes, in order to identify the best molecules in terms of *Activity* value.

A good way to evaluate the goodness of the classification criterion is the confusion matrix, through which we come to the determination of the number of observations classified correctly or incorrectly in their respective classes.

2.2 Definition of the classes

Being in the field of supervised learning, we need to specify how many and how are the classes for this dataset.

ACS Medicinal Chemistry Letter Journal published an article in which the values of the response variable *pIC50 MMP12* are subdivided into 8 classes according to the pharmaceutical company. [5]

As we said in the previous chapter, this variable contains some strings and for this task we want to deal with only numerical values, so considering the variable *Activity* we will treat only 6 classes.

Then we will consider only 1704 molecules whose minimum value for *Activity* is 3.7 while the maximum one is 8 and within this range we will treat only the 6 classes laid down in the article on ACS Medicinal Chemistry Letter Journal.

The distribution of classes among the 1704 molecules is shown in the pie chart in *Figure 1*.

For example, we associate experiments with *Activity* value from 3.7 to 5 (included) to the class A, if the value is between 5 (excluded) and 6 (included) to the class B and so on.

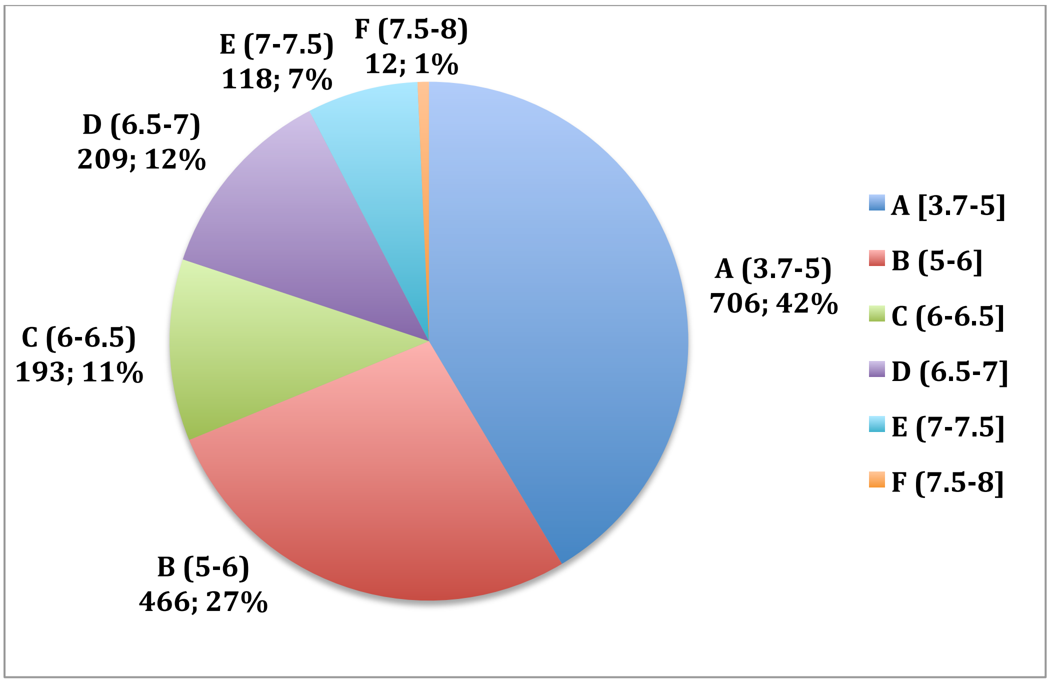


Figure 1: molecular groups

As we could intuit after seeing the its density distribution, we can state that most of the molecules belongs to the class A that contains the lowest values, while the class F which covers the highest values has the lowest number of observations.

2.3 Division in training and test set

Our initial dataset is splitted into training set and test set to obtain unbiased estimates of percentage of correct/incorrect classification.

We have used 500 molecules for the test set (about 30%) while the remaining 1204 observations (about 70%) for the training set.

The training set contains training examples in which each observation is composed by its features and a label representing its class according to the previous definition of classes.

The training set is used for the learning phase to obtain a classifier estimating its parameters.

The test set contains molecules with the same set of measurements but the class membership is not specified and thus it is used to measure the generalization ability of the classifier on data never seen during the learning phase.

The task of the classifier is to assign a class to these observations according to their features.

Evaluating the results obtained, the predictive ability of the model is calculated in the confusion matrix through the percentage of misclassification (number of molecules belonging to the test set that have been assigned to a class different from the real one).

Using the software R, there are 2 steps for each classifier:

- use of the corresponding classification function to estimate the parameters of the model on the training set and the result obtained is stored
- launch of the command *predict* to make predictions on the test set using the estimated classifier

Summarizing, the problem of classification can be divided into 3 phases:

- estimation of classifiers using the training set able to discriminate efficiently between various classes
- calculate the percentage of misclassification using the test set
- choose the most appropriate model according to their generalization ability on new data

2.4 Methods

There are many algorithms for classification but having to deal with also categorical variables will be used the Multinomial Logistic Regression method to test whether *Atag* and *Btag* variables are discriminant in the implementation of the classes.

We have data belonging to C classes.
The matrix X contains the values x_{ij} of the j-th variable for the i-th observation, so i=1,…,n and j=1,…,p.

2.4.1 Multinomial Logistic Regression

The multinomial logistic model is a generalization of the binomial one in the case of classification in more than two groups.
In this classifier, the logit transformation is implemented to ensure that the probabilities belong to a range between 0 and 1.
Then the logistic regression model arises from the desire to model the posterior probabilities of the C classes using linear function in x ensuring that they sum to 1.
We assume that the log-odds of posterior probabilities of the classes can be modelled as a linear function of the inputs:

$$\log\left(\frac{P(G=h|X=x)}{P(G=C|X=x)}\right)=\beta_{h0}+\beta_h^T x \qquad h=1,\dots,C-1$$

The model is defined using $C-1$ log-odds or logit transformations in such a way that the constraint on the sum of the probabilities is valid.
The last class is used as denominator in the odds-ratios of the model but this choice on the reference class is arbitrary.

We can see the result in terms of probabilities of belonging to the classes with a simple calculation:

$$P(G = h | X = x) = \frac{\exp(\beta_{h0} + \beta_h^T x)}{1 + \sum_{l=1}^{C-1} \exp(\beta_{l0} + \beta_l^T x)} \qquad l = 1,...,C-1$$

$$P(G = C | X = x) = \frac{1}{1 + \sum_{l=1}^{C-1} \exp(\beta_{l0} + \beta_l^T x)}$$

They clearly sum to 1 and to stress their dependence on the whole parameter set $\theta = \left\{\beta_{10}, \beta_1^T, ..., \beta_{(C-1)0}, \beta_{C-1}^T\right\}$, we can denote the probabilities as $P(G = h | X = x) = p_k(x;\theta)$.

The parameters of the logistic regression model are estimated via maximum likelihood, using the multinomial distribution for $P(G | X = x)$. [10] [11]

The nnet package provides the function multinom() for the multinomial logit model.

Using the multinomial logistic regression, we have to define the explanatory variables of the model in order to have the minimum percentage of misclassification.

In R, the *stepAIC* function compute an iterative procedure that eliminates variables from the model which are not significant in terms of AIC, using a backwards regression analysis that starts with the initial model composed by all the explicative variables.

The AIC (Akaike Information Criterion) criterion is used to evaluate the model and it contains information both on the goodness of the model and the number of predictors that it contains.

It assumes a high value if the model doesn't fit well or if there are to much predictors, so the goal is to find a model that minimizes the value of AIC.

The variables that can be eliminated are those that lead to a model with a lower AIC value than that of the complete model.

2.5 Results

The results are highly dependent on how I choose the examples to put in the training set and test set which can lead to results absolutely busted.

It is advisable to have in the training set examples belonging to all classes for achieving a good performance of the classifier in terms of correct classifications.

If there are no examples belonging to the all classes in the training set, the classifier hardly will be able to classify observations of the test set belonging to the remaining classes.

For this reason, each method has been applied N times (N=100) then on N different partitions of the training set to have a high probability that we have considered all the 6 classes.

Setting for each run the same seed for all the classifiers, the partitions of the training set and test set are equals and then we are able to determine which is the best algorithm for this dataset which is the method with the smallest mean percentage of misclassification.

For each execution of the classifier, the percentage of misclassification is calculated on the test set from the confusion matrix and making a mean on these values, we get an indication of the goodness of the classifier.

The confusion matrix is a table for evaluating the performance of an algorithm, typically used in supervised learning on the test set.

Each column of the matrix represents the istances in a predicted class, while each row indicates those in an actual group, as we can see in *Table 1* that shows an example.

	A	**B**	**C**	**D**	**E**	**F**
A	209	0	0	0	0	0
B	0	114	11	0	0	0
C	0	0	59	0	0	0
D	0	0	0	70	0	0
E	0	0	0	7	28	0
F	0	0	0	0	2	0

Table 1: example of confusion matrix

In other words, there is a confrontation for each observation of the test set between its real class according to its real value of *Activity* and the class predicted by the classifier.
In this confusion matrix, of the 125 units of actual class B, the classifier predicts that 11 molecules belongs to the class C, and of the 35 of class E, it predict that 7 of them are molecules that fit on class D.
An important aspect is that given the low number of molecules actually belonging to the class F (12 in total on the starting database), the classifier hardly assigns this class to the units of the test set, as succeed in this example.
To try to go beyond this limit, you could organize each training set in such a way that it has a fixed fraction of molecules belonging to each available class (usually 60%-70%).
Looking at the general confusion matrix, it is easy to understand that the main diagonal contains only the units classified correctly while the observations outside from it are taken in account to calculate the error percentage for each execution by dividing the total number of error for the size of the test set (500).
Taking an average of these N percentages, we get a more accurate index on the goodness of the algorithm.

Adding categorical variables in Multinomial Logistic Regression we don't have problems, the mean error percentage varies according with the configuration on the explanatory variables used in the function.
We have considered for the construction of the model the single variables but also the pairwise iterations to understand if there are relationships between the variables involved, as we can see from the *Table 2*.

	% error
Activity	0.128
Activity + Atag	9.02
Activity + Btag	4.49
Activity * Atag	12.05
Activity * Btag	7.82
Activity + Atag + Btag	17.06
Atag	60.86
Btag	39.34
Atag+Btag	30.16

Table 2: error percentage according the model

The best result has been obtained considering *Activity* as the only explanatory variable in each classifier, then taking into account only this variable, the methods estimate the classes for the observations belonging to the test set.
We must consider the fact that this result is excellent because it has been obtained with *Activity* as unique explanatory variable which is the only one that sets out the definition of the classes.
This conclusion regarding the best model among the logistic regression models is confirmed also in terms of AIC and residual deviance using the *stepAIC* procedure starting from the complete model composed by all 3 variables.
Adding other explanatory variables, the mean error percentage increases.
Moreover, using only the categorical variables in the definition of the model, we get a percentage of error of 30.16% and this tells us that these variables have an influence in the definition of the 6 classes determined by the company.
In addiction to predictive purpose, in this case the classification results useful to determine which variables are most influential and useful in determining the class membership.
We can assert that the variable *Btag* has a higher discriminatory capability than *Atag*, taken individually or together with *Activity*, as we have seen with the paired boxplots in the previous chapter.
Another important aspect is that the pairwise interactions are not useful for achieving good results.

After this classification phase, the most interesting conclusions that we can drawn are:

- best results are obtained using only *Activity* as explanatory variable but the classes are defined considering only this variable
- the categorical variables *Atag* and *Btag* have some influence in the definition of the 6 predefined classes
- using the Multinomial Logistic Regression results that *Btag* has a higher discriminant capability rather than *Atag* and the pairwise interactions are not useful in the predictive purpose
- the best model obtained in the Multinomial Logistic Regression is the same using the stepAIC procedure

Most interesting results could be obtained using the similarity matrix that determines how the experiments are similar to each other in the search of clusters among molecules.
This aspect will be discussed in the next chapter that covers the clustering methods.

CHAPTER 3
Clustering approach

3.1 Unsupervised learning: clustering

We said in the previous chapter that in machine learning there is a distinction between supervised and unsupervised learning.

We have already discussed the problem of supervised learning by treating some of the most widely used classification algorithms.

In unsupervised learning the goal is to find the hidden structure and the key features of the data using only unlabelled data, so without the information about the group membership.

The clustering research is one of the approaches to unsupervised learning and it is a multivariate analysis technique.

3.2 Criterions in clustering

Data clustering has gained a lot of attention in many fields such as statistic, data mining, marketing, pattern recognition and bioinformatics.

The task of the cluster analysis is to clumping a set of observations in such a way that the units in the same group (called cluster) are more similar to each other than to those in other groups in order to have homogeneous groups.

Another interpretation on the objective is to find groups with small distances among the cluster members ensuring thus that these observations are similar to each other but at the same time are different respect to those belonging to the other clusters.

Then cluster analysis is a useful method for identifying homogeneous groups of observations in such a way that the objects in a specific cluster share many features but at the same time they are very dissimilar to objects not belonging to that cluster.

The definition of "cluster" is ambiguous in literature and also this has lead to many cluster models with different algorithm implementations but they have in common the aim to create groups of observations separating the data. [12]

Informally, a cluster can be defined using these two criterions:

- internal criterion: all observations inside a cluster should be highly similar to each other so the intra-cluster distances are minimized (coherent group, homogeneity)
- external criterion: all observations outside a cluster should be highly dissimilar to the ones inside so the inter-cluster distances are maximized (external isolation, separation)

3.3 Implementation of clustering algorithm

The starting point is the definition of a measure of distance or dissimilarity between n data points in order to identify clusters.

In clustering algorithms the pairwise distance between each observation of the dataset is defined, depending on the type of data, through a distance metric (Euclidean, Manhattan) or by using an index of dissimilarity which is usually the complementary to 1 of the similarity value. [13]

Observation with smaller distances between one another are more similar, whereas observations with larger distances are more dissimilar.

The dissimilarity matrix or a distance matrix based on a metric has this form (*Figure1*):

$$D = \begin{pmatrix} 0 & d(1,2) & d(1,3) & \dots & d(1,n) \\ d(2,1) & 0 & d(2,3) & \dots & d(2,n) \\ d(3,1) & d(3,2) & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{pmatrix}$$

Figure 1: dissimilarity/distance matrix

where $d(i,j)=d(j,i)$ measures the distance or the dissimilarity between objects i and j.

The definition of this matrix is one of the key issues of cluster analysis because it is the basis for the implementation of clustering algorithms and consequently it has an influence on the obtained results. [14]

Given a set of data objects, a distance/dissimilarity matrix nxn and also the number k of cluster, the classical approach for clustering is:

- define a clustering problem as a partition problem
- run an algorithm that returns the partition in clusters which covers all data
- define a quality measure of the partitioning

3.4 Similarity matrix based on Tanimoto

As said before, the pharmaceutical company provided us a second dataset, collecting a similarity matrix among molecules based on Tanimoto distance.

The values express how one experiment is chemically similar to all the other experiments.

This distance is widely used for molecular fingerprints that is a method of encoding the structure of a molecule.

The most common type of fingerprint is a series of binary digits (bits) that indicates the presence or absence of a particular feature in the molecule.

These comparisons between fingerprints associated to molecules allow to determine the similarity matrix among experiments. [15]

The Tanimoto distance is an extended version of the Jaccard's coefficient similarity and the general formula for calculating is:

$$T(A,B) = \frac{c}{a+b-c}$$

where:

- a is the number of bits turned on (set to 1) in molecule A
- b is the number of bits turned on in molecule B
- c is the number of common bits turned on in both A and B

Simplifying, it is the ratio between the number of features/characteristics common to both objects divided by the sum of features present in A and in B less those in common.

As all similarity measures, it varies between 0 and 1 where 0 means totally different molecules while 1 means maximum similarity and it is a symmetric distance, so $T(A,B) = T(B,A)$.

From this similarity matrix, we can extrapolate the dissimilarity matrix by calculating its complementary matrix, then $Diss = 1 - Sim$. [16]

3.5 Distinction between methods

There are many clustering methods available in literature which have different theoretical aspects and therefore they produce different cluster structures.
The clustering methods used for this analysis can be classified using the following general distinction according to the type of cluster solution that they produce:

- Hierarchical methods: yield an entire hierarchy of clustering of the data using a dissimilarity or distance matrix, producing a nested clustering
- Partitioning methods: divide the dataset into non-overlapping k similar clusters by optimizing a function, where the desired number of cluster k must be specified such that each data object is in exactly one group

At the beginning of the clustering process, we have to select appropriate variables for clustering according to the objective of the analysis because they can provide different segmentations.
It is important to select the variables that provide a clear subdivision of the groups avoiding to use an abundance of explanatory variables.
Each of these clustering procedure follows a different approach to grouping the most similar objects into a cluster. [12] [17]
Now, we will discuss in more detail the features of these two methodologies.

3.6 Hierarchical methods

The hierarchical methods used in this analysis are fully described by Kaufman and Rousseeuw (1990). [18]
The aspect that distinguishes them from non-hierarchical methods is that the assignment of an observation to a cluster is irrevocable, this means that once an object is allocated to a certain cluster, there is no possibility of reassigning this object to another cluster. [12] [13]
This can be considered as a drawback because a bad decision about splitting or grouping objects in one step cannot be corrected in the following steps. [19]

The problem of the construction of the clusters is addressed in two complementary ways, for this reason the hierarchical methods are divided into two sub-classes:

- Agglomerative:

  Starting from N observations as individual clusters, at each step the algorithm merges the closest pair of clusters until only one (or k) clusters left
- Divisive:

  Starting from a unique big cluster containing all the observations, at each step the largest cluster is splitted in two smallest clusters until each one contains an observation (or remain k clusters) [14] [19]

All hierarchical algorithms produce a set of nested clusters organized as a hierarchical tree, called dendrogram that is a graphical tool to see a cluster solution, where ach merge/split is represented by a horizontal line and its y-coordinate is the dissimilarity or distance at which the two clusters are merged/splitted while the clusters are placed along the x-axis.
The same dataset can produce a different sequence of nodes for agglomerative or divisive clusterings on the same dataset. [19]

The interpretation of the dendrogram is immediate and very easy.
In the dendrogram shown in *Figure 2*, observation 3 and 6 are the most similar and they join to form the first cluster, followed by observations 2 and 5.
The last two clusters to form are 3-6-2-5-4 and the single object 1.
Clusters may join pairwise, such as the merging of the observations 3-6 or alternatively, a single observation can be sequentially added to an existing cluster, such as the case of the joining of the cluster 3 with the object 4.
As said before, the strength of clustering is indicated by the level of height (distance/dissimilarity) in the y-axis and in this example the clusters 2 and 3 join at similar levels. [20]

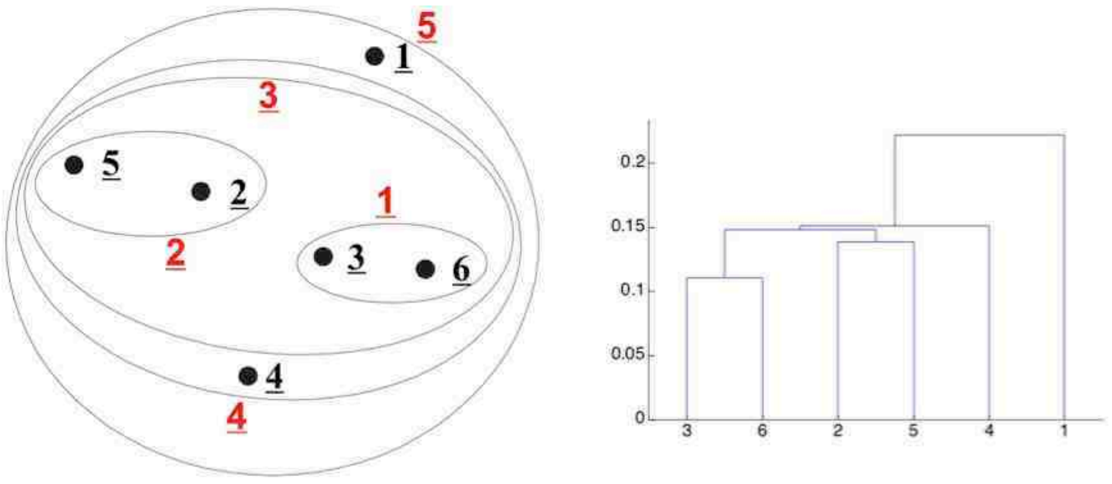


Figure 2: example of dendrogram

Applying a hierarchical algorithm, the number of cluster k is not specified because cutting the dendrogram at the proper level we can obtain any desired number of clusters. [21]

There is a tradeoff between the number k of clusters and the similarity of the elements in each group.

If k is high, the clusters will be small in size end their elements will be highly similar but the analysis on many groups can be difficult.

Instead, if there are few clusters, their larger number of elements will show less similarity to each other, but on the other hand the analysis will be easier. [20]

Sometimes the validity of a cluster is evaluated by considering an index which tells me the strength of the clustering structure found so the amount of structure that has been found by the algorithm. [18]

For agglomerative algorithms there is the agglomerative coefficient AC described by this formula:

$$AC = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \frac{h_i}{h}\right)$$

where h_i is the height in which the i-th observation is reported in its first merged cluster in the dendrogram while h is the maximum height that is the height of the last and only one cluster that is created.

Whereas for divisive algorithms, we consider the divisive coefficient DC following this formula:

$$DC = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{h_i}{h}\right)$$

where h_i is the height in which the i-th observation is reported in its last merged cluster in the dendrogram (before being split as a single object) while h is the maximum height of the whole dataset.

These indexes are between 0 and 1 and the divisive coefficient DC typically assumes slightly larger values when applied to the same dataset.

If the index is very small, we are in a situation in which the algorithm has not found a natural structure because no cluster has been found or the data consists of one big cluster.

On the other hand, the coefficients assume high values close to 1 for dendrograms with a strong clustering which means that a very clear clustering structure has been identified but this does not necessary mean that a good clustering has been found because for example the presence of possible outliers increases the value of these indexes. [14] [18] [22]

3.6.1 Agglomerative algorithms

The initial situation consists in having n separate clusters each of which contains an observation of the dataset.

The observations are connected to form clusters according to their distances which can be of different metrics depending on the available data (for example Euclidean, Manhattan, Mahalanobis) or based on a similarity index.

These quantities are used to compute the distance according to the linkage criterion that determines how to calculate the distances between a new cluster and the existing clusters.

At each step of the algorithm, after calculating the pairwise distances following the linkage criterion using the Lance-Williams dissimilarity update formula, the nearest (most similar) two groups of observations are combined into a higher-level cluster.

Note that we have to merge those observations with the smallest distance, regardless of the clustering procedure and at each step the number of clusters decreases. [12]

Then at the first step, the agglomerative algorithm examines all the distances or dissimilarities between all the clusters and pairs together the two most similar clusters (those with the smallest distance between them) to form a new bigger cluster.

Proceeding in this way, at the last step, all data points are represented by a single big cluster. [12] [18]

In summary, all the hierarchical agglomerative clustering methods can be described by a general algorithm with these steps:

1) starts with N singleton clusters labelled as $1,...,N$ which represent the input data
2) finds the pair of clusters with minimal pairwise distance using the distance function or the dissimilarity matrix according to the linkage criterion
3) joins the two clusters into a larger higher-level cluster, it associates a label to that new cluster and it removes the two old clusters
4) repeats $N-1$ times from step 2 using the Lance-Williams dissimilarity update formula until we obtain an unique big cluster containing all data points

This allows to establish a hierarchy of clusters from bottom-up since the algorithm starts considering N singleton clusters and it ends with a single large cluster. [12]

Linkage criterions

There are several agglomerative procedures and they can be distinguished by the way in which they define the distance from a newly cluster to the others. [12]

A key point in evaluating the results is how the distance between two clusters has been computed because the results on the same dataset depend significantly on this choice.

In fact, different methods not always produce the same hierarchies and this choice on the linkage criterion is reflected on the obtained results. [12] [13]

In hierarchical methods, the main linkage criterions between two groups X and Y in which two observations $x \in X$ and $y \in Y$ are the following:

- Single link

$$d(x,y) = \min_{x \in X, y \in Y} d(x,y)$$

The distance between two clusters is computed as the distance between the two closest elements belonging to these different clusters (*Figure 3*).

In terms of similarity, the two joining groups are those in which two observations have the maximum similarity because the distance is inversely proportional to the similarity concept.

A possible disadvantage is that two clusters may be forced together due to a pair of elements close to each other, even if many of the other observations can be very distant.

In this case, it is necessary that only one pair of elements belonging to the two clusters have a high similarity value because this distance is based only on the two most similar (closest) single points.

It tends to form one large cluster with the other groups containing only few objects. [12] [21]

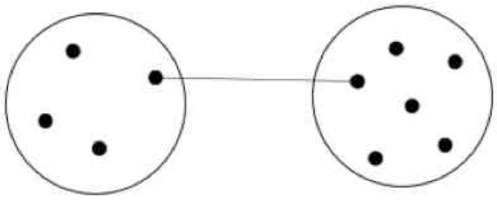


Figure 3: single link concept

- Complete link

$$d(x,y) = \max_{x \in X, y \in Y} d(x,y)$$

The distance between two clusters is computed as the maximal distance between two elements belonging to these different clusters (*Figure 4).*

In terms of similarity, the two joining groups are those in which two observations have the minimum similarity.

Although it is based on two single points, by considering the observations more distant from each other, it avoids the drawback of the single linkage but it can be significantly distorted by moderate outliers.

In this way, this linkage criterion is strongly biased towards finding clusters with approximately equal diameter. [21]

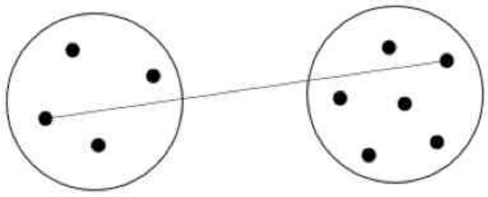


Figure 4: complete link concept

- Average (unweighted pair-group average method UPGMA)

$$d(X,Y) = \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} d(x,y) \qquad \text{with } d(x,y) = |x-y|^2$$

The distance between two clusters is computed as the average distance between all pairs of elements, one in each cluster (*Figure 5)*.

Compared to the previous linkage criterion, it considers all the possible squared pairwise distances by averaging, so it is a compromise between single and complete linkage and it is less susceptible to outliers and noise.

In terms of similarity, the two merged groups are those in which the mean similarity value is the minimum one.

It tends to join clusters with low within-cluster variance and it is slightly influenced towards creating clusters with the same variance and similar size. [12] [21]

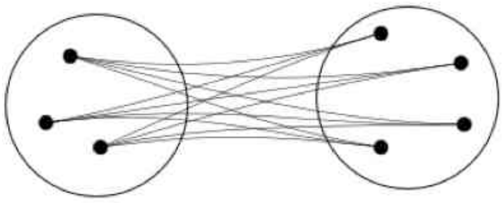


Figure 5: average link concept

- Centroid

$$d(X,Y) = \left\| \overline{c}_X - \overline{c}_Y \right\|^2$$

The distance between two clusters is computed as the squared Euclidean distance between their centroids (*Figure 6)*.

Each cluster is identified by a centroid which is a vector of means of the explanatory variables measured on the observations belonging to that group.

At each step, the distance matrix is recalculated starting not from the previous distances but from the updated centroid of each cluster and the pair of clusters with the most similar centroids is merged.

It is more robust to outliers than most other hierarchical methods but in other cases may not perform as well as them.

It tends to produce clusters with low within-cluster variance and similar sizes, as well as in average linkage. [12] [23]

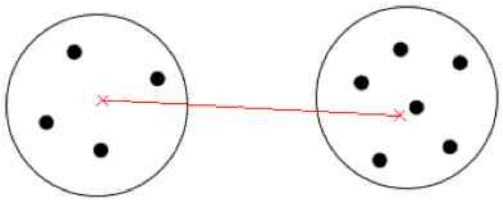


Figure 6: centroid concept

- Median

$$d(X,Y) = \left\| \bar{w}_X - \bar{w}_Y \right\|^2$$

The distance between two clusters is computed as the squared Euclidean distance between their medians.

The median method is similar to the centroid method but the centroids of the two merging clusters are not weighted proportionally to the size of the clusters.

This linkage is better than the previous one if we suspect groups of different sizes. [17]

- Ward

$$d(X,Y) = \sum_{x \in X} \left(x - \bar{c}_X\right)^2 + \sum_{x \in Y} \left(x - \bar{c}_Y\right)^2 - \sum_{x \in Z} \left(x - \bar{c}_Z\right)^2$$

The distance between two clusters is computed as the sum of the within sum of squared distances from the cluster centroids less the within cluster sum of squared distances resulting from the merging of the two clusters.

The two merged groups are those in which occurs the minimum increment in the total within group sum of squares errors thus minimizing the increase of variance within groups (total within-cluster sum of squared errors) because increasing the number of observations contained in a cluster, we have as consequence an increase in the within group variance.
By keeping the distances within the cluster as small as possible, it tends to find clusters with approximately equal size and it is often useful when the other methods find clusters composed by few observations.
It is less susceptible to noise and outliers. [13] [21]

- McQuitty (WPGMA)

$$d(E,C) = \frac{1}{|A||B|}\left(|A| * d(A,C) + |B| * d(B,C)\right)$$

The distance between the new cluster E and another cluster C is computed based on the distances of the two clusters that were merged (A and B to form E) with respect to the cluster C (*Figure 7)*. [24]

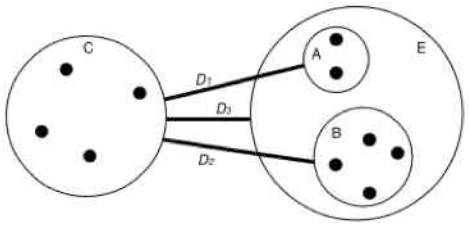


Figure 7: McQuitty concept

Lance-William dissimilarity update formula

The Lance-Williams dissimilarity update formula is a method for determining which is the distance between the new cluster $C_{I,J}$ (C_I and C_J have just been merged to form the cluster $C_{I,J}$) and any other existing clusters C_K

It is useful because all the linkage criterions can be formalized in a unified manner using this formula:

$$d(C_{I,J}, C_K) = \alpha_i d(C_I, C_K) + \alpha_j d(C_J, C_K) + \beta d(C_I, C_J) + \gamma \left| d(C_I, C_K) - d(C_J, C_K) \right|$$

Using only the parameter $\alpha > 0$, the formula becomes:

$$d(C_{I,J}, C_K) = \alpha d(C_I, C_K) + \alpha d(C_J, C_K) + (1-2\alpha) d(C_I, C_K)$$

The formula can be modified by setting the values of the parameters α_i, α_j, β and γ depending on the chosen linkage criterion, as these values are reported in the following table (*Table 1*):

HACM	α_i	α_j	β	γ
Single link	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete link	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Group average	$\frac{m_i}{m_i + m_j}$	$\frac{m_j}{m_i + m_j}$	0	0
Median	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
Centroid	$\frac{m_i}{m_i + m_j}$	$\frac{m_j}{m_i + m_j}$	$-\frac{m_i m_j}{(m_i + m_j)^2}$	0
Ward	$\frac{m_i + m_k}{m_i + m_j + m_k}$	$\frac{m_j + m_k}{m_i + m_j + m_k}$	$-\frac{m_k}{m_i + m_j + m_k}$	0

Table 1: Lance-William dissimilarity update parameters

where m_i is the number of items in C_I. [17]

HCLUST and AGNES algorithms

Two agglomerative algorithms were used in the analysis: HCLUST (Hierarchical CLUstering) and AGNES (Agglomerative NESting).

In R, the functions associated with these algorithms are *hclust* and *agnes*, the latter contained in the library *cluster*.

The function *hclust* performs a hierarchical clustering using a dissimilarity matrix for the observations being clustered and any linkage criterion described above can be used.

Initially, each observation is assigned to its own cluster and then the algorithm proceeds iteratively joining at each step the two most similar clusters, until there is just a unique big cluster.

At each step, the distances between the newly cluster and the existing clusters are recomputed through the Lance-Williams dissimilarity updated formula according to the linkage criterion used.

The function *agnes* constructs a hierarchy of clustering using a data matrix with only numeric variables or a dissimilarity matrix using any linkage criterion except "mcquitty", "centroid" and "median" but in addition you can set the parameters of the Lance-Williams formula using "flexible".

In the latter case, the 4 coefficients of Lance-Williams formula are specified by the vector *par.method* and if it is of length 1 (a), the coefficients are defined as follows: $a_i = a_j = a$, $\beta = 1 - 2a$, $\gamma = 0$.

Initially each observation is a small cluster by itself and at each stage the two most similar clusters are combined to form one larger cluster until only one large cluster remains containing all the observations.

Compared to other agglomerative clustering methods such as *hclust*, *agnes* has more features: it yields the agglomerative coefficient AC which measures the amount of clustering structure found and it also provide the banner plot that is just an alternative graphical tool to the dendrogram. [19] [25] [26]

These 2 functions, being agglomerative algorithm but with different implementation, can lead to the same solution.

As example, we consider a distance matrix of 5 observations (*Table 2*) to show how works a general agglomerative coefficient using the single linkage criterion:

	A	**B**	**C**	**D**	**E**
A	0	1	5	6	8
B	1	0	3	8	7
C	5	3	0	4	6
D	6	8	4	0	2
E	8	7	6	2	0

Table 2: distance matrix of 5 observations

In the first step, the observations A and B are merged having the smallest distance with $d_{AB}=1$.

Now, we need to calculate the distances between this new cluster $\{A,B\}$ and the other observations:

$$d_{(A,B)C} = \min(d_{AC}, d_{BC}) = \min(5,3) = d_{BC} = 3$$

$$d_{(A,B)D} = \min(d_{AD}, d_{BD}) = \min(6,8) = d_{AD} = 6$$

$$d_{(A,B)E} = \min(d_{AE}, d_{BE}) = \min(8,7) = d_{BE} = 7$$

In this way, we obtain a new distance matrix (*Table 3*):

	AB	**C**	**D**	**E**
AB	0	3	6	7
C	3	0	4	6
D	6	4	0	2
E	7	6	2	0

Table 3: updated distance matrix

Again, we have to repeat the previous procedure to calculate the distances among clusters according to the single criterion and so on so forth. [13]

3.6.2 Divisive algorithms

All datasets that can be clustered by means of the agglomerative approach can also be analyzed in a divisive way.

If agglomerative algorithms start with N clusters attributable to the single observations and at each step the closest pair of clusters are merged, with divisive algorithms we have the inverse procedure that is starting from a unique big cluster, at each step there is a split on the biggest cluster.

All the hierarchical divisive clustering methods can be described by a generic algorithm with these steps:

1) starts with N observations labelled as $1,\ldots,N$ into a big cluster
2) finds the largest available cluster

3) splits that cluster into two smaller sub-groups that are homogenous as possible, it labels them and it removes the old cluster
4) repeats $N-1$ times from step 2 until all clusters contain a single object

Then, it is a top-down methodology since the algorithm starts with a single large cluster and it ends with N singleton clusters. [27]

DIANA algorithm

The divisive methods are less commonly used and few algorithms are used.

The divisive algorithm that was used is DIANA (Divisive ANAlysis clustering) and in R its function is *diana*.

This function implements the iterative algorithm proposed by Macnaughton-Smith (1964).

DIANA constructs a hierarchy of clusters using a data matrix with only numeric variables or a dissimilarity matrix starting from a one large cluster containing all the observations.

At each step, the cluster with the largest diameter R, which corresponds to the largest dissimilarity between any two of its observations, is selected and then divided into two clusters A and B until each cluster contains only a single observation.

Initially, A equals R and B is empty and we have to move one object from A to B.

To divide the selected cluster into two new clusters, the algorithm finds the most disparate observation that has the largest average dissimilarity respect to all other observations of the selected cluster, after computing these values for each observation i of A:

$$d\left(i, A \setminus \{i\}\right) = \frac{1}{|A|-1} \sum_{\substack{j \in A \\ j \neq i}} d(i,j)$$

This observation i' initiates the "splinter group" moving it from the original group, so we put:

$$A_{new} = A_{old} \setminus \{i'\}$$

$$B_{new} = B_{old} \cup \{i'\}$$

In the subsequent steps, we look for other points to move from A to B.

The algorithm for each observation of the larger group A compares the average dissimilarity respect the remaining objects in the original cluster and with those in the "splinter group" making their difference:

$$d\left(i, A \setminus \{i\}\right) - d\left(i, B\right) = \frac{1}{|A|-1} \sum_{\substack{j \in A \\ j \neq i}} d(i,j) - \frac{1}{|B|} \sum_{h \in B} d(i,h)$$

After these comparisons, the algorithm detects the observation i'' which lies much further from the remaining objects than from the "splinter group" (maximum difference) and this object moves from A and B.

If the difference is negative for each observation, this means that must not be done other changes and then the division of R into A and B is completed.

The next step is to compute the diameter of the two clusters and that one with the maximum value will be the group to split:

$$diam(Q) = \max_{\substack{j \in Q \\ h \in Q}} d(j,h)$$

and the procedure is repeated until we obtain N clusters.

This value is also used as the level for representing the division in the dendrogram.

The function *diana* provides the divisive coefficient DC that measures the amount of clustering structure found and it also provides the banner plot that is an another graphical tool.

As example, we consider the dissimilarity matrix between 5 observations (*Table 4*) to show how works the function *diana* for the divisive algorithm*:*

	A	**B**	**C**	**D**	**E**
A	0	0.1	0.5	0.9	0.8
B	0.1	0	0.4	0.8	0.7
C	0.5	0.4	0	0.3	0.4
D	0.9	0.8	0.3	0	0.2
E	0.8	0.7	0.4	0.2	0

Table 4: distance matrix of 5 observations

Being divisive, the algorithm assumes that the observations initially form a unique cluster $\{A, B, C, D, E\}$.

In the first step, the algorithm has to split up the cluster into two smaller clusters, looking for the object for which the average dissimilarity respect to all other objects is largest (*Table 5)*:

Object	Average dissimilarity
A	(0.1+0.5+0.9+0.8)/4 = 0.575
B	(0.1+0.4+0.8+0.7)/4 = 0.5
C	(0.5+0.4+0.3+0.4)/4 = 0.4
D	(0.9+0.8+0.3+0.2)/4 = 0.55
E	(0.8+0.7+0.4+0.2)/4 = 0.525

Table 5: average dissimilarity values

Comparing the values, the observation A initiates the splinter group and at this stage we have the groups $\{A\}$ and $\{B, C, D, E\}$.

For each object of the larger group, the algorithm computes the average dissimilarity respect to the remaining objects and those in the splinter group, as shown in *Table 6*:

Object	Avg dissimilarity to original group	Avg dissimilarity to splinter group	Difference
B	(0.4+0.8+0.7)/3 = 0.63	0.1	0.53
C	(0.4+0.3+0.4)/3 = 0.37	0.5	-0.13
D	(0.8+0.3+0.2)/3 = 0.43	0.9	-0.47
E	(0.7+0.4+0.2)/3 = 0.43	0.8	-0.37

Table 6: differences in terms of average dissimilarity

The largest difference is related to object B, which stays much further from the remaining observations in the old group than from the splinter group.

Therefore B changes membership becoming a member of the splinter group, which now consists of the objects $\{A, B\}$ while the remaining group is $\{C, D, E\}$.

Repeating the procedure calculating the average dissimilarities respect the two groups, we obtain: (*Table 7)*

Object	Avg dissimilarity to original group	Avg dissimilarity to splinter group	Difference
C	(0.3+0.4)/2 = 0.35	(0.5+0.4)/2 = 0.45	-0.1
D	(0.3+0.2)/2 = 0.25	(0.9+0.8)/2 = 0.85	-0.6
E	(0.4+0.2) = 0.3	(0.8+0.7)/2 = 0.75	-0.45

Table 7: differences in terms of average dissimilarity

At this step, all the differences are negatives because the average dissimilarity values respect the splinter group is greater and then no further changes must be made.
This is the first divisive step completed which splits the dataset into the clusters $\{A,B\}$ and $\{C,D,E\}$.

In the next step, the algorithm divides the biggest cluster (group with the largest diameter that is the highest dissimilarity value between two of its objects) following the previous procedure. [18]

3.7 Partitioning methods

Using partitioning methods a completely different approach is adopted.
As said before, in a partitional clustering there is a division of the dataset into non-overlapping similar clusters such that each observation is in exactly one cluster (mutually exclusive) and in this case it is necessary to establish the number k of clusters that we desire in the cluster solution. [21]
The initialization phase consists in choosing a center for each of the k clusters and the method initially allocates the observations to the cluster that is closest by some distance measure.
The partition algorithm iteratively adjusts the cluster centers to best fit the associated observations until no further improvement can be made. [28]
In fact, the observations are reallocated to the nearest representative cluster during the iterative procedure, producing thus a single partition according to the optimization of a certain criteria.

In contrast to hierarchical algorithms, at each step the obtained partition is being questioned by considering new cluster centers around which aggregate the observations, so the assignment of an observation to a cluster is not irrevocable and this means that the observations are reassigned to a different cluster if the initial allocation is inappropriate.

There are several algorithms which differ in these aspects:

- how the cluster centers are initialized
- how the elements are assigned to the clusters
- how some or all elements can be reassigned to a different group [13]

Some of the most commonly used partition clustering methods have been applied for the analysis and they are k-means, PAM (Partition Around Medoids) and fuzzy clustering.

PAM is less sensitive to outliers and noise than k-means but they have in common the property that they provide "hard clusters", meaning that each observation is assigned to only one cluster.

An alternative to this approach is fuzzy clustering where ach observation is assigned to every cluster according to a degree of membership. [28]

3.7.1 K-means algorithm

K-means is one of the simplest partitioning methods in its implementation.

This algorithm segments the dataset in k homogenous clusters in such a way that the within cluster variance (within-group sum of squares) is minimized.

It employs a greedy iterative approach to find a good clustering solution and it can converge to a locally optimal solution.

The first step in the clustering process is to choose, often randomly, a centroid for each cluster that is the mean vector of the numerical variables of the observations in that group:

$$m_i = \frac{\sum_{i \in C_i = i} x_i}{N_i} \qquad i = 1, \ldots, k$$

In each step of the iterative process, each observation is assigned to the cluster with the closest centroid minimizing the within-cluster variation, which is basically the squared distance from each observation to the cluster centroid:

$$C_i = \arg\min \sum_{i=1}^{k} \sum_{x \in C_i} \left\| x - m_i \right\|^2$$

where x is an observation in cluster C_i and m_i is the centroid of the cluster C_i and each observation is bint to the cluster with the minimum distance.

After the assignments of the observations to the k clusters, the centroids are recalculated shifting into positions and the algorithm repeats the iterative assignments procedure considering the new centroids until the convergence is achieved (centroids don’t change then the allocations remain the same) or a predetermined number of iterations is reached.

An important aspect is that in contrast to the hierarchy methodology, an allocation to a cluster can change in the course of the clustering process.

Most of the convergence happens in the first few iterations but sometimes the stopping condition is changed in “Until relatively few points change clusters”.

It is very important the initialization phase where we have to choose the initial centroids because sometimes they are readjusted in the right way during the iterative procedure but this don’t happens always, thus arriving at different conclusions.

K-means has problems when the data contains outliers.

In addition, the k-means algorithm can have limitations when clusters are of differing sizes and densities. [12] [21] [29]

General algorithm:

```
- select $k$ centers as the initial centroids
- *repeat*
        for all observations compute the distances to the $k$ centroids
        assign all objects to the closest centroid
        recompute the centroid of each cluster
- *until* the centroids don’t change or a predetermined number of iterations is reached
```

As example, we consider 2 explanatory variables of 7 observations (*Table 8*) to show how works the function *kmeans* using k =2*:*

Observation	Variable 1	Variable 2
A	1	1
B	1.5	2
C	3	4
D	5	7
E	3.5	5
F	4.5	5
G	3.5	4.5

Table 8: data of 7 observations

In the initialization phase, we have to randomly choose the 2 centroids for the clusters and in this case they are: m_1 =(1.5,1.7) and m_2 =(5.2,7.4).

Now we have to allocate the observations to the closest cluster according to the previous formula for calculating the distance.

If we consider the observation B we have these distances:

$$d(A,m_1)=\sqrt{(1-1.5)^2+(1-1.7)^2}=0.86$$

$$d(A,m_2)=\sqrt{(1-5.2)^2+(1-7.4)^2}=7.65$$

then we associate this element to cluster 1 and repeating the calculation for the others observation, we have this situation (*Table 9*):

Observation	Distance centroid 1	Distance centroid 2	Nearest cluster
A	0.86	7.65	1
B	0.3	6.54	1
C	2.74	4.05	1
D	6.35	0.45	2
E	3.86	2.94	2
F	4.46	2.5	2
G	3.44	3.36	2

Table 9: nearest cluster for each observation

Thus, the two clusters are composed by $\{A,B,C\}$ and $\{D,E,F,G\}$ having new centroids:

$$m_1 = \left(\frac{1}{3}(1+1.5+3), \frac{1}{3}(1+2+4)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5+3.5+4.5+3.5), \frac{1}{4}(7+5+5+4.5)\right) = (4.12, 5.37)$$

The iterative procedure goes on until the centroids don't change or a predetermined number of iterations is reached.

The function associated in R is *kmeans* that can be implemented using 4 different versions formulated by Hartigan-Wong (1979), Forgy (1965), Lloyd (1982) and MacQueen (1967).

The default algorithm is Hartigan-Wong and this is considered to be the most robust and generally does a better job respect the others but it is often recommended trying several random starts.

In the Forgy's version, the cluster centroids are recomputed after all data points have been assigned, while in MacQueen's version, after the allocation of each observation and not at the end of a cycle of reallocation, the centroid of the cluster that has gained the element and that one that has lost the point are recalculated.

Moreover, another difference is that the Forgy's method iterates until converged while the MacQueen's basic algorithm performs only one complete pass through data.

In addition, the starting points of the MacQueen's algorithm are often the first k observations in the dataset. [28] [30] [31]

The R function *kmeans* uses only a data matrix with only numeric variables and imposing the initial configuration of k centroids the 3 versions converge to a unique solution.

3.7.2 PAM algorithm

The PAM algorithm (1987) is based on the search for k representative objects called medoids among the observations of the dataset, one for each cluster and it requires that all the variables are of quantitative type.

In other words, a medoid is the most centrally located object in a cluster.

The algorithm considers possible choices for the medoids and after finding them, each element is assigned to the nearest representative object thus forming the k clusters.

PAM algorithm is more insensitive to outliers and noise than k-means but it is more computationally expensive.

As example, let's consider the dataset containing 7 objects each characterized by two variables, given in *Table 10* and supposing that the data must be divided into k =2 clusters, we consider the observations 4 and 8 as the selected medoids:

Observation	**Variable 1**	**Variable 2**
A	1	4
B	5	1
C	5	2
D	5	4
E	10	4
F	25	4
G	25	6

Table 10: data of 7 observations

In *Table 11* the Euclidean distances from each observation to the two medoids and its nearest cluster are given:

Observation	**Distance object 1**	**Distance object 5**	**Minimal distance**	**Closest medoid**
A	0	9	0	1
B	5	5.83	5	1
C	4.47	5.38	4.47	1
D	4	5	4	1
E	9	0	0	2
F	24	15	15	2
G	24.08	15.13	15.13	2
			6.23	

Table 11: nearest medoid for each observation

The average distance in this case is 6.23 and this value gives an indication about the quality of the clustering considering the strength of the clusters.

There is an algorithm that is able to select k medoids in such a way to have a very low average dissimilarity/distance and so a good partition solution.

In this way, the medoid is an object of the cluster for which the average dissimilarity/distance respect to the other observations of the cluster is minimal.
The algorithm used in PAM minimizes the sum of these dissimilarities which is mathematically equivalent to minimize the average dissimilarity of objects to their closest medoid but the calculations result more accurate.

The algorithm consists of two steps:

- BUILD phase: iterative selection of k representative objects creates an initial clustering
- SWAP phase: attempt to improve the set of medoids and consequently the cluster solution changing the medoids

BUILD phase

The first selected medoid is the one for which the sum of the dissimilarities to all other observations is smallest.
At each step, another medoid is selected and it is that one that decreases the objective function as much as possible, executing these phases:
- consider an observation i that has not yet been selected as medoid
- consider another non-selected object j and calculate the difference between its dissimilarity D_j respect the most similar medoid and its dissimilarity $d(j,i)$ with observation i
- if this difference is positive, the observation j contributes to the decision to select object i as medoid calculating:

$$C_{ji} = \max(D_j - d(j,i), 0)$$

- calculate the total gain obtained by selecting the element i using the formula:

$$\sum_j C_{ji} \qquad \text{(objective function)}$$

- select as medoid the observation i that minimizes the total gain:

$$\min_i \sum_j C_{ji}$$

This process ends when the algorithm finds k medoids, one for each cluster.
The algorithm assigns each observation to the cluster with the nearest medoid.

SWAP phase

In this second phase, we have to consider all pair (i,h) for which only object i has been selected as medoid:

$$i \in \{m_1,...,m_k\} \text{ and } h \notin \{m_1,...,m_k\}$$

The role of the two objects is exchanged (h is a medoid) and the effect of this change is evaluated considering the value of a clustering determined by k representative objects, that is defined as the sum of dissimilarities between each observation and the most similar medoid.
This second phase goes on until no exchange leads to an improvement in the clustering solution in terms of total distance.
To calculate the effect of this swap between i and h on the clustering value, the algorithm executes these two steps:

1) consider a non-selected object j and calculate its contribution C_{jih} to the swap between object i and h:
   a) if j is more distant from both i and h than its medoid k (*Figure 8)* then the contribution is null:

$$C_{jih} = 0$$

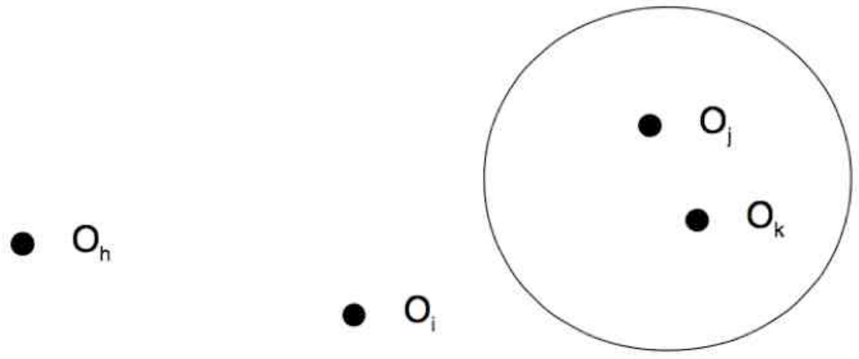


Figure 8

The object k is the medoid of j.

b) if j is not further from its medoid i $\left(d(j,i) = D_j\right)$ than from any other medoid, there are two possible situations:

- if j is closer to h than to the second closest medoid $j2$ (*Figure* 9) then the contribution of object j to the swap between elements i and h is:

$$d(j,h) < d(j,j2) \quad \rightarrow \quad C_{jih} = d(j,h) - d(j,i)$$

C_{jih} can be positive or negative depending on the positions of the elements j, h, and i: C_{jih} is positive if object j is closer to i than to h $\left(d(j,i) < d(j,h)\right)$ then the swap is not favorable regarding j

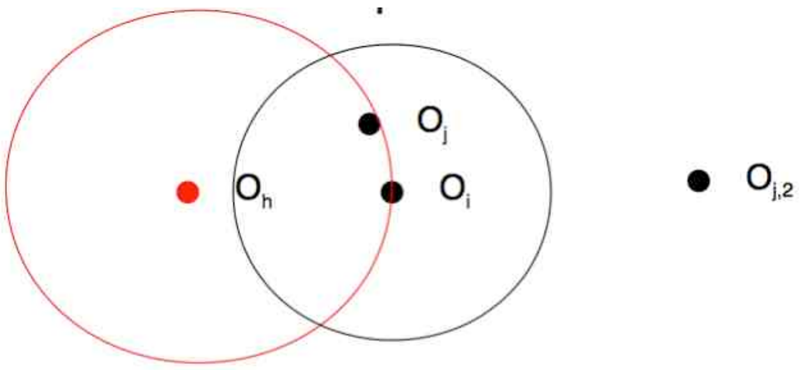


Figure 9

The object j is allocated to the cluster with medoid h.

- if j is at least as distant from h than from the second closest medoid $j2$ (*Figure* 10) then the contribution of object j to the swap is:

$$d(j,h) \geq d(j,j2) \quad \rightarrow \quad C_{jih} = d(j,j2) - d(j,i)$$

C_{jih} can be only positive and the swap is not suggested

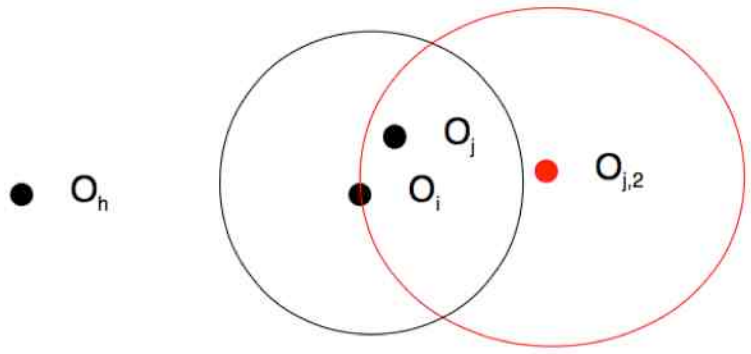


Figure 10

The object j is bind to the cluster with medoid $j2$.

c) if j belonging to the cluster with k as medoid is more distant from i respect to at least one of the other medoid but it is closer to h than to any medoid (*Figure* 11), then the contribution of j to the swap is:

$$C_{jih} = d(j,h) - d(j,k)$$

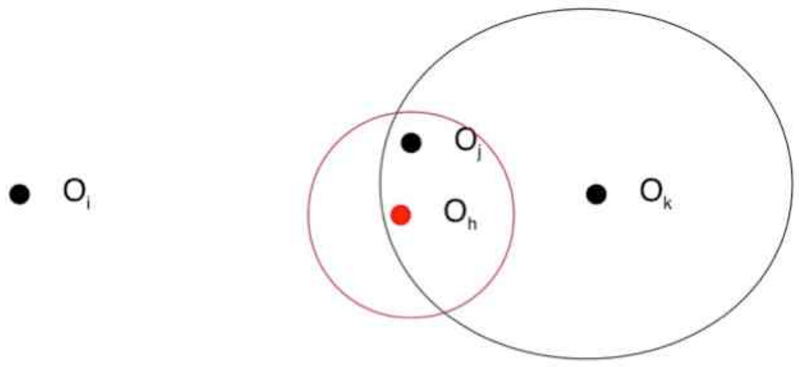


Figure 11

The object j is allocated to the cluster with medoid h.

2) calculate the total result of the swap by adding the contributions C_{jih}:

$$T_{ih} = \sum_j C_{jih}$$

3) to decide if the swap must be done, select the pair (i,h) that:

$$\min_{i,h} T_{i,h}$$

if $\min T_{i,h}$ is negative, the swap is carried out $(i \leftrightarrow h)$ and the algorithm returns to step 1, whereas if it is positive or equal to 0 then the swap is not suggested and the algorithm stops.

A graphical representation of the clustering solution is provided by means a silhouette plot (Rousseeuw 1987) that gives an indication of the quality of the clustering with a certain number of clusters k.

As example, *Figure 12* shows the silhouette plot for the iris dataset considering k =3 also because the observations really belong equally to 3 classes.

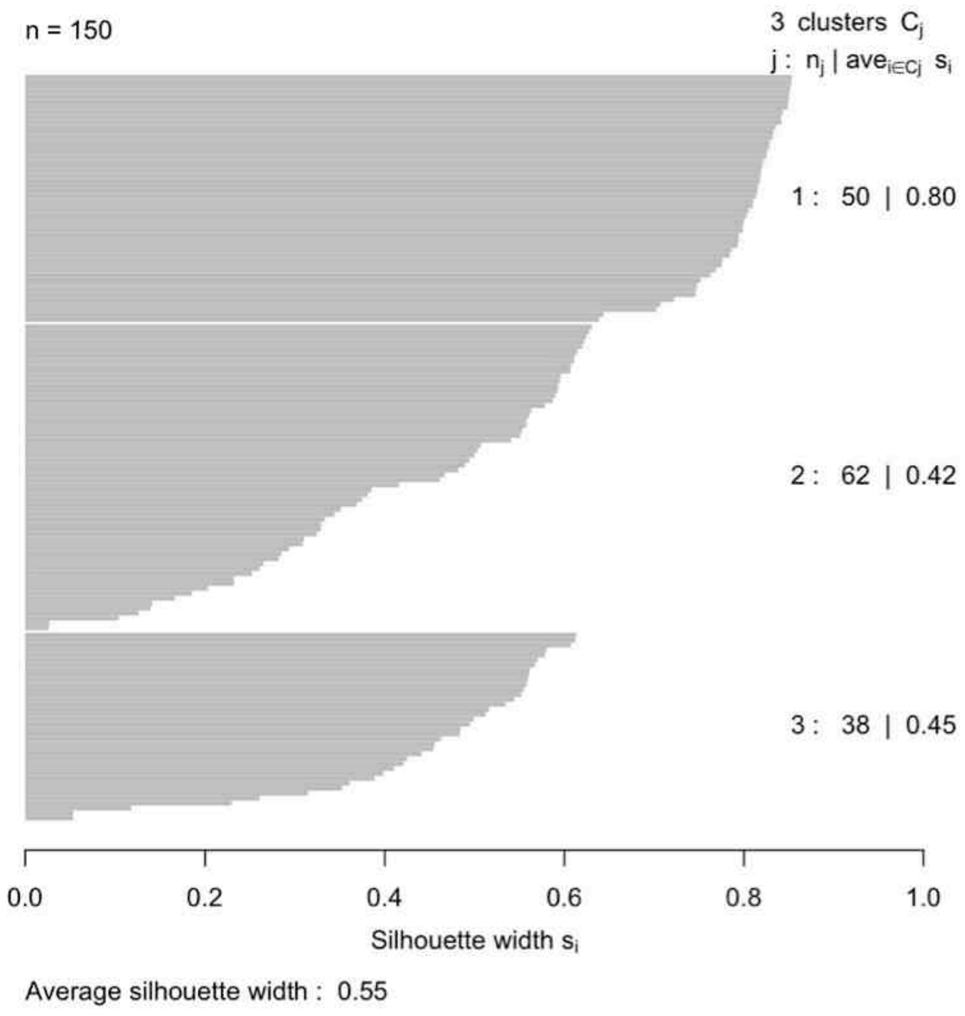


Figure 12: silhouette plot

Each cluster is represented by a silhouette showing how each observation lies within a particular group according to the silhouette width value, allowing thus a comparison about the quality of each cluster.

Each observation i is associated with one cluster, generally called A, and then the silhouette width value $s(i)$ is defined using the notions of:

$$a(i) = \frac{1}{|A|-1} \sum_{j \in A, j \neq i} d(i,j)$$

average dissimilarity of i respect all other observations within cluster A

$$d(i,C) = \frac{1}{|C|} \sum_{j \in C} d(i,j)$$

average dissimilarity of i respect all observations contained in any cluster C different from A

$$b(i) = \min_{C \neq A} d(i,C)$$

minimum average dissimilarity considering a cluster C different from A

The cluster B for which the minimum is achieved, so $b(i) = d(i,B)$, is called the *neighbor* of the observation i and it can be considered as the second-best choice for the object i.

Then the value of $s(i)$ is obtained by combining the previous quantities in this way:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \qquad -1 \leq s(i) \leq 1$$

Note that if the cluster A has only a single observation, we simply set $s(i)$ =0.

If $s(i)$ is close to 1 then the object i is "well classified" in cluster A and this implies that the within dissimilarity $a(i)$ is much smaller than the $b(i)$ value.

In this case we are pretty sure that the observation has been assigned to an appropriate cluster and the neighbor cluster B is not so close to i as the actual group A.

If $s(i)$ is close to 0, the assignment of the object i to clusters A or B is not clear because the $a(i)$ and $b(i)$ values are approximately equal, so it can be considered as an intermediate case.

If $s(i)$ is close to -1, the object i stays on average much closer to cluster B than to A because $a(i)$ value is much larger than $b(i)$, then it would be more correct to assign the element i to cluster B concluding that i has been misclassified.

The silhouette plot displays the silhouette width value $s(i)$ for each observation i respect to its cluster A, ranked in decreasing order.

The average of the $s(i)$ for all observations belonging to a particular cluster is called the *average silhouette width* of that cluster and looking at these values we are able to distinguish clear clusters from weak ones.

In *Figure 11*, on the right is reported for each cluster the number of observations associated to that cluster and its average silhouette width.

Considering the average of $s(i)$ for all observations of the dataset, we obtain the *average silhouette width for the entire dataset* $\overline{s}(k)$ that gives a global indication about the clustering solution quality using k clusters.

Taking the maximum value of $\overline{s}(k)$ over all possible values for k (k = 2,3,…,n-1), we have the *silhouette coefficient*:

$$SC = \max_k \overline{s}(k)$$

This index gives us a suggestion about the optimal value of k and consequently a measure of the amount of clustering structure that has been discovered by the algorithm. [18] [26] [28] [30]

A possible interpretation of the silhouette coefficient SC is summarized in *Table 12*:

SC	Proposed interpretation
0.71-1	found a strong structure
0.51-0.7	found a reasonable structure
0.26-0.5	found a week structure
< 0.25	no substantial structure found

Table 12: interpretation of silhouette coefficient SC

In R the function associated to PAM is simply *pam* but it is necessary to load the library *cluster*.
In constructs a partition of clusters using a data matrix with only numeric variables or a dissimilarity matrix and it gives the ability to impose the initial configuration of k medoids by choosing k observations from the dataset.

3.7.3 Fuzzy clustering

In many practical situations, we need clusters that overlap but by definition applying a partition algorithm the groups are disjoint.
As said previously, an alternative to the mutually exclusive approach is fuzzy clustering where ach observation is assigned to every cluster according to a degree of membership. [28]
A fuzzy clustering technique is then useful to describe situations in which an object lays at approximately the same distance among some clusters.
For example, a fuzzy clustering algorithm is able to say that an object belongs mainly to a certain cluster or should be divided almost equally between two clusters.
This degree of membership to each cluster is the membership coefficients and its range is between 0 and 1.
Then a fuzzy method spreads each observation over the various clusters, so for each object i and each cluster v there is a percentage of membership u_{iv} that indicates how strongly object i belongs to cluster v or in other terms how each object fits into each group.

As example, let's consider a dataset of 5 objects on which has been applied a fuzzy clustering method with k =3, obtaining a list of membership coefficients (Table 13):

Object	Membership		
	Cluster 1	**Cluster 2**	**Cluster 3**
1	0.87	0.06	0.07
2	0.93	0.03	0.04
3	0.06	0.86	0.08
4	0.10	0.10	0.80
5	0.35	0.42	0.23

Table 13: percentage memberships for 3 clusters

The interpretation of this table is very easy: observation 1 belongs for the most part to the cluster 1 because it has 87% of membership while observation 5 is almost equally distributed among the three clusters, so it can be considered as an intermediate case but it is closer to cluster 2.
Obviously the sum of the membership coefficients for each object must be equal to 1 (100%).
Having the percentages of membership to each cluster, this algorithm shows more information on the structure of the data.
The FANNY (Fuzzy Analysis) algorithm iteratively minimizes the following objective function:

$$\sum_{v=1}^{k} \frac{\sum_{i,j=1}^{n} u_{iv}^2 u_{jv}^2 d(i,j)}{2\sum_{j=1}^{2} u_{jv}^2}$$

where u_{iv} indicates the unknown membership of observation i to cluster v, $d(i,j)$ is the distance or dissimilarity between objects i and j and the membership exponent r is 2.
The sum in the numerator ranges over all pairs of objects $\{i,j\}$, so each pair $\{i,j\}$ is encountered twice because $\{j,i\}$ also occurs and this is the reason of the division by 2.

The membership quantities u_{iv} have these following constraints:

$$u_{iv} \geq 0 \qquad \text{for } i = 1,\ldots,n;\ \ v = 1,\ldots,k;$$

$$\sum_{v} u_{iv} = 1 \qquad \text{for } i = 1,\ldots,n;$$

The algorithm finds these u_{iv} values by minimizing the objective function until the convergence is achieved.
The minimization of the objective function is carried out using an iterative procedure that uses Lagrange multipliers (see pag. 182 [18])

Some fuzzy clustering solutions are more fuzzy than others: we have complete fuzziness when each observation has the same percentage of membership in all clusters (hence $1/k$), whereas if each object is allocated to a certain cluster with a percentage equal to 1 (hence 0 percentage in the other clusters), we are in the case of hard clustering (mutually exclusive) as in partition clustering.
The Dunn's partition coefficient F_k (1976) measures how hard a fuzzy clustering is:

$$F_k = \sum_{i=1}^{n} \sum_{v=1}^{k} \frac{u_{iv}^2}{n}$$

Its minimum value is $1/k$ occurs in the case of completely fuzzy clustering (all $u_{iv} = 1/k$, whereas when the algorithm acts as a partitioning methods (all u_{iv} = 0 or 1), the index assumes its maximum value which is 1.
The normalized version of this index is given by:

$$F_k^{'} = \frac{F_k - (1/k)}{1 - (1/k)} = \frac{kF_k - 1}{k - 1}$$

It is easy to interpret it because it assumes values in the range between 0 and 1.
From a fuzzy clustering it is possible to obtain the closest hard clustering by assigning each observation to the cluster associated with the highest percentage of membership.
Considering the previous example, observations 1 and 2 are clearly related to cluster 1, while observation 3 to cluster 3.

However, the intermediate observations must be allocated then observation 5 is associated with cluster 2 because its percentage (42%) is larger with respect to that of its second cluster (35%).
Also in fuzzy clustering, the silhouette plot described in PAM can be used to evaluate the fit of objects within a particular group according to the silhouette width value considering only the dissimilarities, allowing thus a comparison about the quality of each cluster. [18] [26] [28]

The fuzzy clustering algorithm implemented in R is present in the library *cluster* and its function associated is *fanny.*
It uses a data matrix with only numeric variables or a dissimilarity matrix and it has an important parameter *memb.exp* r that specifies the membership exponent used in the objective function and it can be reduced to obtain a less fuzzy membership of objects in each cluster.
Furthermore, useful for this analysis is the closest hard clustering where for each object returns the cluster that corresponds to the highest percentage of membership.

3.8 Dissimilarity using *daisy*

We are also interested to verify whether the categorical variables *Atag* and *Btag* are significant for identifying clusters that respect as much as possible the composition of the 6 classes.
A dissimilarity matrix can be computed in R using two different functions depending on the data present in the dataset: *dist* uses, as we have seen, only quantitative variables while the main feature of *daisy* is that it works with other types of variables such as nominal, ordinal, binary (a)symmetric and ratio scaled variables.
Applying the function *daisy* to *Atag*, *Btag* and *Activity* variables, the dissimilarity values take into account two categorical variables and a numerical one.
The ability of handling this mixture of variables is obtained by using a generalization of Gower's dissimilarity coefficient (1971):

$$d_{ij} = d(i,j) = \frac{\sum_{l=1}^{p} w_k \delta_{ij}^{(l)} d_{ij}^{(l)}}{\sum_{k=1}^{p} w_k \delta_{ij}^{(l)}} \qquad \in [0,1]$$

In other words, the dissimilarity value between two observations is the weighted mean of the contributions of each variable, specifically d_{ij} is a weighted mean of $d_{ij}^{(l)}$ with weights $w_l \delta_{ij}^{(l)}$, where:

- w_l : weight for each variable (default: 1 in Gower's original formula, so each variable has the same importance)
- $\delta_{ij}^{(l)}$: weight is 0 when the l-th variable is missing in either or both objects (i and j) otherwise it is 1 so this weight indicates the presence of the variable
- $d_{ij}^{(l)}$: l-th variable contribution to the total dissimilarity, specifically, if the variable is categorical, it assumes value equal to 0 if both values are equal (same modality) 1 otherwise, but if the variable is quantitative, its contribution is given by the absolute difference of both values divided by the total range of the variable:

$$d_{ij}^{(l)} = \frac{|x_{il} - x_{jl}|}{\max_h x_{hl} - \min_h x_{hl}}$$

The dissimilarity values d_{ij} range between 0 and 1 since each individual contribution $d_{ij}^{(l)}$ is in [0,1].

The output from *daisy* is a dissimilarity object that can be used as input for the algorithms described above. [14] [18] [26] [32]

As example, we consider *Atag*, *Btag* and *Activity* components of two molecules to calculate their dissimilarity value according to Gower's coefficient:

Atag	Btag	Activity
A01	B01	6.5
A01	B10	5.6

First of all, we calculate the contribution to the total dissimilarity of the quantitative variable *Activity* that must be added to the formula:

$$d_{ij}^{(l)} = \frac{|6.5 - 5.6|}{8 - 3.7} = 0.209$$

Therefore, the dissimilarity value between these two molecules results:

$$d_{ij} = d(i,j) = \frac{1*1*0+1*1*1+1*1*0.209}{1*1+1*1+1*1} = \frac{1+0.209}{3} = 0.403$$

3.9 Results of clustering solutions

We are interested in determining whether the use of the similarity matrix based on Tanimoto distance between the objects to be clustered leads to a meaningful splitting of the molecules into groups.

We would like to have significant groups in the clustering solution that only makes use of the information contained in the similarity matrix but it is often difficult to spot a structure in a dataset by merely looking at its similarity matrix.

Similarity value $s(i,j)$ takes values between 0 and 1 but it is necessary to compute its complement to 1 because the clustering algorithms use a dissimilarity matrix as a distance matrix to gradually build clusters.

It can be shown that there is no absolute "best" criterion for the evaluation of a clustering solution, consequently a specific criterion must be supply in such a way that the result will suit the needs as much as possible.

Only the 1704 molecules with a numeric value for *pIC50 MMP12 - Activity* and their similarity values have been considered in the clustering process because as comparison criterion of the clustering solution has been used how these observations are really distributed among the 6 classes based on *Activity* values, seen in the previous chapter (*Table 14*).

	A [3.7-5]	B (5-6]	C (6-6.5]	D (6.5-7]	E (7-7.5]	F (7.5-8]	
Freq.	706	466	193	209	118	12	1704
%	41,43	27,35	11,33	12,27	6,92	0,70	100

Table 14: composition of classes

Then for each of these observations there is a comparison between its real class and its cluster of membership established by the clustering algorithm, using as number of clusters k = 6.

In this context, the criterion to evaluate the quality of a clustering solution is to allocate each cluster formed by the algorithm to a dominant class that is the class of membership of the majority of observations which belong to that cluster.
It is preferable that each cluster is associated to a different dominant class in order to have these 6 clusters related to all the classes.
It is very difficult that a clustering algorithm, especially if it uses a dissimilarity matrix, is able to produce clusters in which each of these clusters contains molecules belonging to only one class.

An example of an optimal solution is reported in *Table 15* where each column of the matrix shows the percentages of instances predicted in a particular cluster $(1,\ldots,6)$ distributed to the various classes ($A,\ldots,F$), so we have to analyze this table by column:

%	**1**	**2**	**3**	**4**	**5**	**6**
A	0	0	96	0	0	0
B	95	0	4	0	0	0
C	5	0	0	0	0	97
D	0	98	0	0	0	3
E	0	2	0	0	100	0
F	0	0	0	100	0	0

Table 15: distribution of the clusters

For example looking at the first column, we can say that cluster 1 has 95% of observations belonging to class B while the remaining 5% are observations of class C.

Moreover, clusters 4 and 5 contain only molecules of only one class, respectively class F and class E.

Note that the sum of each column is 100, so each column shows the distribution of the observations of the cluster among the 6 classes.

Then in this case, each cluster is associated with a different majority class according to its highest percentage (*Table 16*).

cluster	**1**	**2**	**3**	**4**	**5**	**6**
class	B	D	A	F	E	C
%	95	98	96	100	100	97

Table 16: majority classes

On the other hand, it's also important to see how the observations of a particular class are distributed among the clusters created by the algorithm because it makes no sense to allocate a cluster to a majority class if the cluster contains only few observations of that class.

As we mentioned earlier, 95% of observations in cluster 1 are of class B but in what proportion corresponds taking into account all the 466 molecules belonging to class B?

Considering the previous example, a table is used for this purpose (*Table 17*) but in this case each column shows the percentages of molecules owned by a class ($A,...,F$) among the clusters ($1,...,6$) formed by the clustering algorithm:

%	**A**	**B**	**C**	**D**	**E**	**F**
1	0	97	8	0	0	0
2	0	0	0	90	5	0
3	100	3	0	0	0	0
4	0	0	0	0	0	100
5	0	0	0	0	95	0
6	0	0	92	10	0	0

Table 17: distribution of the classes

For example looking at the second column, we can say that most of observations of class B is inside of cluster 1 (97%) while the remaining 3% belongs to cluster 3.
Note that also here, the sum of each column is 100, so each column shows the distribution of the molecules of a class among the 6 clusters.
Then, it is necessary to consider both tables (*Table 15*, *Table 17*) to evaluate the goodness of a solution.
Table 18 summarizes the previous matrices reporting for each cluster the two percentages relative to its dominant class:

	1	**2**	**3**	**4**	**5**	**6**
	B	D	A	F	E	C
%in cluster	95	98	96	100	100	97
% in class	97	90	100	100	95	92

Table 18: summary cluster-dominant class

We look at the first column to better understand this table: cluster 1 is associated to class B because 95% of molecules in this cluster are of class B but in the evaluation must be considered that this quantity of molecules corresponds to 97% of the total observations of class B.
Note that in this case, both tables (*Table 15*, *Table 17*) have the highest percentage in each column located in correspondence of the same pair cluster-class.
The solution taken as example reported in *Table 18* can be considered optimal because each cluster is associated to a different class and the percentages are rather high.
Concluding, for each clustering solution provided by any possible configuration of an algorithm previously described, following this proposed procedure of evaluating, we want to verify if there is this matching between the composition of each cluster with that of the class associated.

Following the goal of the clustering procedure for this project, we already know the number of segments that have to be derivated from the dataset, which is $k = 6$ because we have 6 classes based on *Activity* value.

In the R functions used to apply partitioning methods, it is necessary to specify the number k of clusters but this is not the case for hierarchical methods because this is the main distinction between these two approaches.
The function *cutree* has been used in hierarchical methods to assign molecules to a defined number of clusters in such a way that each case fall into one of the 6 non-overlapping clusters.
It takes as parameter the dendrogram of the clustering solution along with a number specifying the number of clusters k that you want to split the tree into.
Practically, the dendrogram is cut such that k distinct subtrees are left.

There are a lot of combinations that the described algorithms can provide by changing parameters according to the algorithm used.
Summarizing foregoing, the parameters that can be set in the corresponding R functions are the following:

- dissimilarity matrix used for the analisys:

  $1 - sim(i,j)$

  $sim(i,j) * mean(i,j)$

  $\left(1 - sim(i,j)\right) * mean(i,j)$

  $dist(i,j) * mean(i,j)$

  $dist(i,j) * sim(i,j)$

  $dist(i,j) * \left(1 - sim(i,j)\right)$

  distance in term of Activity values
  dissimilarity computed with daisy

  $daisy(i,j) * mean(i,j)$

  $daisy(i,j) * sim(i,j)$

  $daisy(i,j) * \left(1 - sim(i,j)\right)$

  where:

  $sim(i,j)$ = similarity value between molecules i, j

  $dist(i,j)$ = difference between *Activity* values of molecules i, j

$mean(i,j)$ = mean between *Activity* values of molecules i,j

$daisy(i,j)$ = dissimilarity of molecules i,j computed with *daisy*

this aspect will be discussed later in this chapter

- all the described algorithms have as parameter a dissimilarity matrix or a data matrix (in this case *Activity* variable) using as metric the Euclidean distance but there are some exceptions: *hclust* can only use a dissimilarity matrix while *kmeans* utilizes a data matrix
- linkage criterions in *hclust*, *agnes*
- *kmeans* algorithm has 4 different versions (Hartigan-Wong, Lloyd, Forgy, MacQueen) and it gives the ability to set the initial centroids as well as in *pam* with its initial medoids
- *par.method* in *agnes* is used for the calculation of Lance-Williams dissimilarity update formula: they proposed taking a value of alpha slightly larger than 0.5 so that $1-2\alpha$ becomes a small negative number where α is a strictly positive constant
- *memb.exp* (r) in *fanny* function specify the membership exponent used in the objective function and it can be reduced to obtain a less fuzzy membership of objects in each cluster (default: r =2 because it can leads to complete fuzzyness) but in the analysis has been mainly considered the hard clustering assignment case where each output cluster reflects the highest percentage of membership

If an algorithm uses a dissimilarity matrix, this must be converted into a distance matrix using the function *as.dist* that practically does not consider its main diagonal and its upper or lower part.

On the other hard, if a data matrix is used, each row corresponds to an observation and each column is a numeric variable and considering the context of the project, the unique variable that can be used is *Activity*.

Moreover, in this case the Euclidean distance (*metric=Euclidean*) is used to calculate the distances between molecules for the creation of clusters.

Usually in *kmeans* the initial centroids are chosen randomly but it is possible to establish the initial configuration of k centroids and a good idea is to choose them

as central values of the k classes (for example, class B varies between 5 and 6, so one of the 6 initial centroids is 5.5).
In this way, initially the molecules are mapped to their closest clusters according to their distances from these centroids and using any of the 4 versions we obtain the same solution.
Similarly, in *pam* is possible to set the initial configuration of the 6 medoids but in this case, the closest molecule to the central value of each class is considered as a medoid (for example, class B varies between 5 and 6, so the nearest molecule to the *Activity* value equal to 5.5 becomes a medoid).
In *fanny*, the parameter *memb.exp* has been also reduced but the hard clustering assignment has been mainly considered where each output cluster reflects the highest percentage of membership for each observation.
All these possible settings have been applied to the data provided by Glaxo in order to obtain the best possible result using the methods described previously.
We are interested to verify whether the similarity matrix based on Tanimoto distance leads to clusters that fit together with the compositions of the 6 classes.
To apply the algorithms useful for this purpose (*hclust*, *agnes*, *diana, pam, fanny*), the similarity matrix must be converted into a dissimilarity matrix computing its complement to 1 and then as a distance matrix (*as.dist* function).
Using this dissimilarity matrix in the various algorithms, not significant results have been achieved because for almost all clusters created by the algorithm result that A is the dominant class, that is these clusters are formed mainly by molecules of class A.
This is due to the values that the similarity matrix assumes, as we have noted in chapter 1: the highest density is in the range between 0.2 and 0.4 approximately (91.16% of similarity values) and higher values occur when two molecules share the same *Atag* or *Btag* component.
As consequence, we have to do mainly with low similarity values thus with molecules not so much similar chemically to each other and then it is difficult to create clusters that are at the same time internally homogeneous but different among them.
This similarity matrix has been used in combination with the new created mean matrix in which the mean between each pair of molecules is calculated trying to

create new dissimilarity matrices $sim(i,j)*mean(i,j)$, $(1-sim(i,j))*mean(i,j)$) that are able to create clusters near to our needs.

There are improvements but also using these matrices, significant results have not been achieved because in most clusters created by the algorithm result that A is the dominant class.

The variable *Activity* has been used as data matrix in the algorithms (*agnes*, *diana*, *kmeans*, *pam* and *fanny*) instead of dissimilarity matrix to verify if these algorithms, based only on this variable, are able to achieve good solutions computing Euclidean distances

As might be expected, by setting the initial configuration of the 6 centroids in *kmeans* as central values of the classes and using any version, each cluster, except the first, contains molecules belonging to 2 adjacent classes whereas the molecules of each class, unless class F, are distributed over two clusters, as we can see from *Table 19* and *Table 20*:

%	1	2	3	4	5	6
A	100	41.81	0	0	0	0
B	0	58.19	72.32	0	0	0
C	0	0	27.68	51.53	0	0
D	0	0	0	48.47	51.04	0
E	0	0	0	0	48.96	66.67
F	0	0	0	0	0	33.33

Table 19: distribution of the clusters

%	A	B	C	D	E	F
1	72.52	0	0	0	0	0
2	27.48	57.94	0	0	0	0
3	0	42.06	38.86	0	0	0
4	0	0	61.14	53.11	0	0
5	0	0	0	46.89	79.66	0
6	0	0	0	0	20.34	100

Table 20: distribution of the classes

Looking at *Table 19*, although initially the molecules are mapped to their closest cluster according to their distances from the centroids, all clusters except the first centroid, contain a substantial part of molecules that don't belong to their corresponding centroid (for example cluster 3 has as centroid the value equal to 6.25 but 72.32% of its observations is related to molecules of class B).

Applying the algorithm *agnes* with the linkage criterion "average", we get these results presented in *Table 21* and *Table 22*:

%	**1**	**2**	**3**	**4**	**5**	**6**
A	0	0	88.33	0	100	0
B	0	0	11.67	100	0	0
C	81.78	0	0	0	0	0
D	18.22	57.44	0	0	0	0
E	0	40.83	0	0	0	0
F	0	1.73	0	0	0	100

Table 21: distribution of the clusters

%	**A**	**B**	**C**	**D**	**E**	**F**
1	0	0	100	20.57	0	0
2	0	0	0	79.43	100	41.67
3	62.18	12.45	0	0	0	0
4	0	87.55	0	0	0	0
5	37.82	0	0	0	0	0
6	0	0	0	0	0	58.33

Table 22: distribution of the classes

Cluster 4-5-6 contain molecules of a single class (100%) thus are associated respectively with classes B, A and F.

The observations of these classes are present in these clusters respectively with the percentages of 87.55%, 37.82% and 58.33% then only cluster 4 has a high percentage.

Most observations of cluster 1 are of class C (81.78%) for which it becomes the majority class representing all the 193 observations then this association results significant.

By the same reasoning, cluster 2 is associated with class D (57.44%), which corresponds to 79.43% of all molecules of class D.

At this point remains cluster 3 that contains observations belonging to class A (88.33%) and class B (11.67%) but they are already dominant classes in clusters 5 and 4

Considering that this cluster is almost entirely made up of molecules belonging to class A (88.33%) and that these molecules are distributed mainly in this cluster (62.18%), it is worth considering this cluster and to bind it to class A.

In this way, the only class that remain is E and a possible solution is to allocate this class to cluster 2 (40.83%) since it contains observations of the adjacent class D so that this cluster also contains the totality of molecules of class E (100%).

Table 23 summarizes what has been said about the association cluster-dominant class:

	1	**2**	**3**	**4**	**5**	**6**
	C	D+E	A	B	A	F
% in cluster	81.78	57.44 + 40.83	88.33	100	100	100
% in class	100	79.43 + 100	62.18	87.55	37.82	58.33

Table 23: summary cluster-dominant class

A difference of 0.1 can be related to two molecules whose *Activity* values are both low (for example 4 and 4.1) or between molecules with high values (for example 7.2 and 7.3) and the Euclidean distance does not give this information.

In order to take into account this additional information, a new dissimilarity matrix has been created by combining the mean matrix with the new formed matrix of Euclidean distances that contains the pair-wise differences between molecules according to their *Activity* values ($dist(i,j) * mean(i,j)$).

An important aspect is that if two molecules have the same Activity value then their difference is set by default to 0.01 in order to take account their relationship if we combine this difference with another value, as in this case.

This dissimilarity matrix is converted in a distance matrix using the function *as.dist* and applied to algorithms useful for this purpose (*hclust, agnes, diana, pam, fanny*).

Applying the algorithm *agnes* with the linkage criterion "flexible" using *par.method=0.8*, we get these results presented in *Table 24* and *Table 25*:

%	**1**	**2**	**3**	**4**	**5**	**6**
A	0	0	0	58.72	0	100
B	0	0	0	41.28	100	0
C	81.78	0	0	0	0	0
D	18.22	100	0	0	0	0
E	0	0	90.77	0	0	0
F	0	0	9.23	0	0	0

Table 24: distribution of the clusters

%	**A**	**B**	**C**	**D**	**E**	**F**
1	0	0	100	20.57	0	0
2	0	0	0	79.43	0	0
3	0	0	0	0	100	100
4	36.26	38.63	0	0	0	0
5	0	61.37	0	0	0	0
6	63.74	0	0	0	0	0

Table 25: distribution of the classes

Proceeding in the same way, clusters 2-5-6 contain molecules of a single class (100%) thus are related respectively to classes D, B and A.
These molecules represent, respect the total number of observations for each class, respectively the percentages of 79.43%, 61.37% and 64.74%.
Looking at the highest values, cluster 1 is associated with class C (81.78%) so that it contains all molecules of this class (100%) and cluster 3 to class E (90.77%) having within all its molecules (100%).
At this point we are in the same situation as before, it remains cluster 4 that contains observations belonging to class A (58.72%) and class B (41.28%) but they are already dominant classes in clusters 6 and 5.
Although the percentages are not very high, this cluster can be bind to its dominant class A (58.72%) and the only class that remains is F that can be allocate to cluster 3

(9.23%) since it contains observations of the adjacent class E so that this cluster also contains the totality of molecules of class E (100%).

Table 26 summarizes these associations also reporting their percentages:

	1	**2**	**3**	**4**	**5**	**6**
	C	D	E + F	A	B	A
% in cluster	81.78	100	90.77 + 9.23	58.72	100	100
% in class	100	79.43	100 + 100	36.26	61.37	63.74

Table 26: summary cluster-dominant class

After all these analysis, the best result has been obtained using this dissimilarity matrix based on means and distances applied to *diana* algorithm, as we can see from *Table 27* and *Table 28*:

%	**1**	**2**	**3**	**4**	**5**	**6**
A	0	0	12.22	0	100	0
B	22.48	0	87.78	0	0	0
C	55.62	0	0	0	0	0
D	21.9	66.5	0	0	0	0
E	0	33.5	0	91.07	0	0
F	0	0	0	8.93	0	100

Table 27: distribution of the clusters

%	**A**	**B**	**C**	**D**	**E**	**F**
1	0	16.74	100	36.36	0	0
2	0	0	0	63.64	56.78	0
3	7.65	83.26	0	0	0	0
4	0	0	0	0	43.22	41.67
5	92.35	0	0	0	0	0
6	0	0	0	0	0	58.33

Table 28: distribution of the classes

Clusters 5-6 contain molecules of a single class (100%) thus are easily associated respectively with classes A, F.

The observations of these classes are present in these clusters respectively with the percentages of 92.35% and 58.33%.

As regard the other clusters, we proceed in the same way by assigning them to the classes with the highest percentage.

Cluster 1 can be bind to the dominant class C (55.62%) but it is a cluster of difficult assignment because it contains significant percentages of molecules belonging to three different classes even though there are no alternatives because all the molecules of class C are within cluster 1 (100%).

For the other three remaining clusters, the percentages of the dominant classes are quite significant.

Table 29 summarizes these associations also reporting their percentages:

	1	**2**	**3**	**4**	**5**	**6**
	C	D	B	E	A	F
% in cluster	55.62	66.5	87.78	91.07	100	100
% in class	100	63.64	83.26	43.22	92.35	58.33

Table 29: summary cluster-dominant class

As we have seen commenting the previous clustering solutions, this table is easy to interpret.

An important aspect that has not been encountered in previous solutions is that each cluster is allocated to a different dominant class and in this case there is an association 1-to-1 between cluster and class.

For example, the best group is 5 because it is entirely composed by molecules of class A (100%) which corresponds to 92.35% of all the 706 observations of this class.

On the other side, cluster 1 is for the most part formed by molecules of class C (55.62%) but it contains the totality of molecules of this class (100%).

For this reason, the proposed procedure for evaluating a solution considers the distributions of both clusters and classes, trying to maximize these two aspects in order to achieve a good clustering solution.

Analyzing the molecules present in each cluster, the observations that are outside the dominant class are those with *Activity* values in proximity of the extremities of the range of the class.
In the following table are reported for each cluster, in addition to the dominant class, the number of molecules and its average value of *Activity* (*Table 30*):

	1	2	3	4	5	6
	C	D	B	E	A	F
n°	347	200	442	56	625	7
mean	6.29	7	5.36	7.39	4.42	7.84

Table 30: cluster summary

All the dissimilarity matrices used for the analysis can be normalized in order to have values that vary between 0 and 1 as in the case of the similarity matrix based on Tanimoto distance, using the following formula:

$$I_{\min} < I_i < I_{\max} \rightarrow \textit{normalization} \rightarrow I^n = \frac{I_i - I_{\min}}{I_{\max} - I_{\min}}; 0 < I^n < 1$$

Applying these normalized dissimilarity matrices are obtained the same results achieved using the matrices without this transformation (except with *agnes* algorithm in the case of changing the parameter *par.method* using the metric *dist*mean*).
In addition, creating the dissimilarity matrix that contains the pair-wise distances between molecules in terms of *Activity*, we have the same results achieved using the Euclidean metric as distance, but being a dissimilarity matrix it is possible to use also the *hclust* algorithm for further analysis.

Comparing the results obtained using this dissimilarity matrix based on *daisy* with those achieved considering only the pairwise Euclidean differences between *Activity* values of molecules, occurs a worsening in the clustering solutions.
This leads to say that *Atag* and *Btag* variables are not useful in the formation of the 6 clusters similar to the composition of the classes.
In addition, changing the weights associated to the 3 variables, it is seen that by giving more importance to *Btag* variable than *Atag*, the results are not yet good but

there is a general improvement in the clustering solutions and this means that *Btag* is more discriminant in the formation of the clusters, as demonstrated in the previous chapter about the classification.

After carrying out these analysis with the proposed evaluation method, we can draw the following conclusions:

- to evaluate a clustering solution have been considered the distributions of the clusters with respect to classes and vice versa, trying to find a matching between the highest percentages
- the similarity matrix based on Tanimoto distance is not useful for this analysis because often it returns A as majority class for each cluster
- unsatisfactory results are obtained also multiplying the similarity/dissimilarity matrix with other matrices
- it is difficult to find associations 1-to-1between clusters and classes, especially for class F that represents only 12
- better results are obtained by considering also the mean between 2 molecules into the dissimilarity matrix
- the best result is achieved by applying the algorithm *diana* using the dissimilarity matrix $dist(i,j)*mean(i,j)$
- combining the distance matrix in terms of *Activity* with the mean matrix have been achieved better solutions than those obtained using the same distance matrix with similarity/dissimilarity values and this confirms what has been said about the poor usefulness of the similarity matrix based on Tanimoto distance in the formation of the required clusters
- using *daisy* for computing dissimilarities we have seen that *Atag* and *Btag* variables are not useful in the formation of the 6 clusters similar to the composition of the classes but *Btag* is more discriminant

3.10 Some highlights from clustering

After analyzing the obtained results using clustering methods, we want to intuit if *Atag* and *Btag* variables and similarity values based on Tanimoto distance are not

useful in identifying clusters as close as possible to the composition of the 6 classes, as it has been said commenting the results.
The following table (Table 31) shows for each class, how many different *Atag* and *Btag* components are present within the class and the corresponding tag with the highest numerosity, reporting also its percentage.

A (706) [3.7-5]		**B (466) (5-6]**		**C (193) (6-6.5]**		**D (209) (6.5-7]**		**E (118) (7-7.5]**		**F (12) (7.5-8]**	
Atag	**Btag**	**Atag**	**Btag**	**Atag**	**Btag**	**Atag**	**Btag**	**Atag**	**Btag**	**Atag**	**Btag**
46 A38/44 3.54%	38 B46 4.81%	45 A37 5.58%	46 B10 6.44%	43 A17 4.66%	22 B23 8.29%	45 A40 4.31%	20 B02 10%	36 A41 9.32%	15 B18 22.03%	10 A21 16.7%	4 B07/25 33.3%

Table 31: predominant Atag and Btag components in each class

For example, class A consisting of 706 molecules whose range is from 3.7 to 5, contains 46 different *Atag* components and 3.54% is the percentage of the components A38 and A44 which are those more present inside the class, while regarding the *Btag* component, there are 38 different modalities and B46 is the predominant one having the highest percentage with 4.81%.
Considering each class, the number of different *Atag* and *Btag* components is high but making a comparison between the two numbers, there are many fewer *Btag* modalities (except for class B) and this aspect can confirm the intuition that *Btag* variable is more discriminant than *Atag*, as mentioned in the previous chapter on classification issues.
These comparisons should be made by considering the numerosity of each class, especially for class F that is composed by only 12 molecules.
Given the high numbers about *Atag* and *Btag* modalities and the percentages rather low also for predominant modalities in each class, it can be stated that these variables are not very useful for identifying the 6 clusters close to the requests.

Regarding the usefulness of the similarity matrix based on Tanimoto distance, for each class have been considered the pairwise similarity values of each its molecules compared to other observations belonging to the same class (Table 32):

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
A	0.143	0.254	0.285	0.303	0.325	0.821
B	0.144	0.259	0.293	0.309	0.333	0.827
C	0.179	0.283	0.317	0.337	0.362	0.814
D	0.176	0.298	0.333	0.352	0.378	0.867
E	0.208	0.333	0.372	0.399	0.435	0.843
F	0.313	0.367	0.404	0.444	0.498	0.71

Table 32: similarity values distribution for each class

For example, class A is composed by 706 molecules for which it was created a matrix of size 706x706 containing all the pairwise similarity values in order to obtain information about its density distribution.

Each similarity density distribution, but especially mean and median indices, is compared with that of the similarity matrix based on all 2500 molecules (Table 33), seen in chapter 1 concerning the preliminary descriptive analysis.

Min	1st Qu.	Median	Mean	3rd Qu.	Max	sd
0.1429	0.2564	0.2887	0.3020	0.3271	0.8667	0.075

Table 33: summary similarity values

We have already said that the highest density is in the range between 0.2 and 0.4 approximately (91.16% of similarity values) and in general the mean and median indices of the distributions fall in the middle of this range, although it should be noted that the indices related to classes E and F are more close to the right tail of the overall distribution of similarity values.

Then we can say that the molecules belonging mainly to classes A, B, C, and D are not particularly chemically similar to each other, so we don't have homogeneous groups.

This aspect allows us to conclude that the similarity matrix based on Tanimoto distance is not particularly discriminant in the formation of the 6 clusters as close as possible to the compositions of the classes.

It may be helpful if there were only E and F classes that contain molecules more similar to each other.

Eucludean distance of Activity values

agnes(average)
4 → B(100%) → 87.55%
6 → F(100%) → 58.33%
5 → A(100%) → 37.82%
1 → C(81.78%) → 100%
2 → D(57.44%)+*E(40.83%)*
→ 79.43%+*100%*
3 → A(88.33%) → 62.18%

diana
5 → A(100%) → 63.74%
4 → E(87.63%)+*F(12.37%)*
→ 72.03%+*100%*
2 → D(83.42%) → 79.43%
6 → B(80.5%) → 34.55%
1 → C(78.17%) → 79.79%
3 → B(54.37%) → 65.45%

agnes(single)
2 → D(100%) → 32.54%
6 → F(100%) → 16.67%
5 → E(89.47%) → 72.03%
1 → C(65.6%) → 42.49%
4 → A(55.03%)+*B(36.32%)*
→ 100%+100%
3 → D(74.81%) → 46.89%

agnes(flexible, par.method=0.6)
5 → A(100%) → 56.52%
6 → B(100%) → 41.63%
4 → E(80.95%)+*F(19.05%)*
→ 43.22%+100%
2 → D(59.39%) → 46.89%
1 → C(56.76%) → 100%
3 → A(56.54%) → 43.48%

DIST*MEAN

diana (best result)
5 → A(100%) → 92.35%
6 → F(100%) → 58.33%
4 → E(91.07%) → 43.22%
3 → B(87.78%) → 83.26%
2 → D(66.5%) → 63.64%
1 → C(55.62%) → 100%

agnes(flexible, par.method=0.8)
2 → D(100%) → 79.43%
6 → A(100%) → 63.74%
5 → B(100%) → 61.37%
3 → E(90.77%)+*F(9.23%)*
→ 100%+100%
1 → C(81.78%) → 100%
4 → A(58.72%) → 36.26%

agnes/hclust(average)
4 → A(100%) → 63.74%
6 → F(100%) → 58.33%
2 → E(95.93%) → 100%
5 → B(75.41%) → 49.36%
1 → D(63.91%)+*C(36.09%)*
→ 100%+*61.14%*
3 → A(52.03%) → 36.26%

hclust(complete/mcquitty)
agnes(complete/weighted)
4 → B(100%) → 73.61%
5 → A(100%) → 45.47%
6 → F(100%) → 58.33%
1 → C(63.49%) → 100%
2 → E(53.39%)+*D(44.34%)*
→ 100%+*46.89%*
3 → A(75.79%) → 54.53%

hclust(median)
4 → B(100%) → 73.61%
6 → A(100%) → 45.47%
5 → E(80.95%)+*F(19.05%)*
→ 43.22%+*100%*
1 → C(63.49%) → 100%
2 → D(59.39%) → 46.89%
3 → A(75.79%) → 54.53%

hclust(centroid)
4 → A(100%) → 92.35%
6 → F(100%) → 50%
5 → B(63.84%) → 42.06%
1 → C(51.9%) → 42.49%
2 → D(51.75%)+*E(45.91%)*
→ 63.64%+*100%*
3 → B(83.33%) → 57.94%

agnes(flexible, par.method=0.6)
5 → F(100%) → 100%
6 → A(100%) → 37.82%
4 → B(91.28%) → 87.55%
2 → E(71.52%) → 100%
1 → D(51.27%)+*C(48.73%)*
→ 77.51%+*79.79%*
3 → A(88.33%) → 62.18%

DIST*MEAN NORM

agnes(flexible, par.method=0.7)
5 → A(100%) → 92.35%
3 → B(84.96%) → 65.45%
4 → E(80.95%)+*F(19.05%)*
→ 43.22%+*100%*
2 → D(59.39%) → 46.89%
1 → C(58.11%) → 79.79%
6 → B(80.5%) → 34.55%

agnes(flexible, par.method=0.8)
2 → D(100%) → 77.51%
5 → A(100%) → 72.52%
6 → B(100%) → 34.33%
1 → C(84.28%) → 100%
3 → E(66.67%)+*F(6.78%)*
→ 100%+*100%*
4 → B(58.19%) → 57.94%

DISTDISS (Distance as dissimilarity)
DISTNORM (Normalized distance)

hclust(mcquitty)
5 → A(100%) → 63.74%
6 → F(100%) → 58.33%
4 → E(94.44%) → 72.03%
3 → B(72.04%)+*C(27.96)*
→ 61.37%+57.51%
1 → D(64.51%) → 100%
2 → A(58.72%) → 36.26%

agnes(average) - only DistNorm
4 → A(100%) → 92.35%
6 → F(100%) → 58.33%
3 → B(84.96%) → 65.45%
1 → C(58.11%) → 79.79%
2 → E(53.39%)+*D(44.34)*
→ 100%+46.89%
5 → B(80.5%) → 34.55%

agnes(average) - only DistDiss
4 - 5 → A(100%) → 62.18%, 37.82%
3 → B(100%) → 100%
6 → F(100%) → 58.33%
1 → C(81.78%) → 100%
2 → D(57.44%)+*E(40.83%)*
→ 79.43%+*100%*

agnes(flexible, par.method=0.6)
5 - 6 → A(100%) → 30.45%, 61.9%
3 → B(88.84%) → 92.27%
4 → E(80.95%)+*F(19.05)*
→ 43.22%+*100%*
2 → D(59.59%) → 46.89%
1 → C(56.76%) → 100%

DIST*SIM

hclust(complete)
agnes(complete)
5 → A(100%) → 56.52%
6 → B(100%) → 41.63%
4 → E(80.95%)+F(19.05%)
→ 43.22%+100%
2 → D(66.5%) → 63.64%
1 → C(63.28%) → 100%
3 → A(56.54%) → 43.48%

hclust(average)
agnes(average)
5 - 6 → A(100%) → 62.18%, 37.82%
4 → F(100%) → 100%
3 → B(100%) → 100%
1 → C(71.75%) → 100%
2 → D(52.99%)+E(47.01%)
→ 63.64%+100%

hclust(median)
5 → A(100%) → 37.82%
6 → B(100%) → 42.06%
4 → E(80.95%)+F(19.05%)
→ 43.22%+100%
1 → C(63.49%) → 100%
2 → D(59.39%) → 46.89%
3 → A(61.92) → 62.18%

hclust(centroid)
5 → A(100%) → 37.82%
6 → F(100%) → 16.67%
2 → D(61.03%) → 79.43%
4 → E(54.55%) → 10.17%
1 → B(45.37%)+C(44.68%)
→ 42.06%+100%
3 → A(61.92%) → 62.18%

agnes(flexible, par.method=0.6)
5 → A(100%) → 56.52%
6 → B(83.4%) → 42.06%
1 → C(78.17%) → 79.79%
2 → D(61.03%) → 79.43%
4 → F(50%)+E(50%) → 100%+10.17%
3 → A(53.21%) → 43.48%

diana
5-6 → A(100%) → 30.45%,42.07%
4 → E(87.63%)+F(12.37%)
→ 72.03%+100%
2 → D(83.42%) → 79.43%
3 → B(63.94%) → 73.82%
1 → C(53.91%) → 100%

DIST*(1-SIM)

hclust(mcquitty)
agnes(weighted)
4 → B(100%) → 56.87%
6 → A(100%) → 37.82%
5 → E(87.63%)+F(12.37%)
→ 72.03%+100%
2 → D(86.36%) → 100%
1 → C(71.22%) → 100%
3 → A(78.11%) → 62.18%

hclust(median)
2 → B(100%) → 74.89%
4-5 → A(100%) → 69.55%,30.45%
3 → E(80.95%)+F(19.05%)
→ 43.22%+100%
1 → D(65.93%) → 100%
6 → C(56.51%) → 78.76%

diana
2 → D(100%) → 100%
6 → A(100%) → 63.74%
5 → B(100%) → 48.71%
3 → E(90.77%)+F(9.23%)
→ 100%+100%
1 → C(51.6%) → 100%
4 → A(81.53%) → 36.26%

CHAPTER 4
Clustering methods based on Tanimoto similarity

4.1 Introduction

We focus on two of the most widely used clustering algorithms in the pharmaceutical industry that can use the similarity matrix based on Tanimoto distance as criterion for the formation of clusters.

Both algorithms are unsupervised non-hierarchical methods but the have different features.

4.2 Jarvis-Patrick algorithm

J-P clustering algorithm (1973) is similar to nearest-neighbors (knn) approach because it is based on similarity between neighbors.

The main concept of this algorithm is that if two different molecules share enough common nearest neighbors, then these two molecules are in the same cluster.

The neighborhood between two molecules is defined by their similarity value and a high value represents two molecules close together.

It requires a matrix containing the pair-wise distances (similarity matrix) and two user-defined parameters that are:

- Number of neighbors to examine: j

  It indicates how many nearest neighbors for each object must be considered for enumerates common neighbors with other molecule.

  Low values lead to the formation of few and small clusters while we have the opposite situation using a high value.
- Minimum number of shared nearest neighbors: k

  It indicates the minimum number of shared nearest neighbors that two object must have in such a way these two molecules belong to the same cluster.

  Obviously it must not exceed than the previous parameter and low values lead to compact clusters, instead using high values the clusters are more dispersed.

The basic procedure is composed by these simple steps:

- For each item find its j nearest neighbors in order to have its neighbors list
- Two molecules belong to the same cluster if:
  - they are included in each other's neighbor list
  - at least their respective k nearest neighbor list match

It is easy to deduce that the clustering solution is highly influenced by the choice of the two parameters because the procedure can produce clusters which are at the same time large and heterogeneous or homogeneous but too small. [33] [34] [35]
In fact, the mean similarity within a cluster generated by the algorithm can vary significantly according to the values of these two parameters.
This aspect about the quality of the clusters is considered as a drawback from many researcher and it reduces the appeal towards this algorithm.

4.3 Butina algorithm

The main feature of Butina algorithm (1999) is the formation of homogeneous clusters where their similarities reflect the Tanimoto index used for the clustering procedure.
In other words, clusters generated from this algorithm have analogous within similarity which is very near to the Tanimoto value used as threshold.
This algorithm is based on the concept of cluster centroid which is the most similar item compared to every other molecule within the cluster, given a Tanimoto similarity value.
To identify such cluster centroids, the number of neighbors is calculated for each molecule at the Tanimoto level chosen as threshold.
Given a molecule and a Tanimoto value used as threshold, a neighbor is found if a similarity value of this target molecule respect to an other is at least equal to the threshold.
Then, a cluster centroid is the molecule within a cluster which has the highest number of neighbors because it is the element most similar to the other members of the cluster.

The paper written by Butina [36] describes the procedure using these steps:

- *Generation of Fingerprints*

For each molecule are generated Daylight's fingerprints encoded as an ASCII string of 1's and 0's using a fingerprinting algorithm

Fingerprints give a considerable amount of information about the structural features of a molecule because each pattern of the molecule is encoded as a set of bits (usually 4 or 5 bits/pattern).

- *Computation of Tanimoto similarity*

The Tanimoto similarity value between two molecules is computed using the usual formula.

- *Identification of potential cluster centroids*

The procedure to find potential cluster centroids has already been described above.

The set is sorted in descending order in such a way the molecules with the highest number of neighbors are placed at the top.

This is the computationally most expensive step.

- *Creation of clusters at a given Tanimoto level*

Starting from the element in the sorted list that becomes centroid c_0, the algorithm consider all those molecules with a Tanimoto value above or equal the threshold, making them members of that cluster.

Each of these molecules that becomes a member of the given cluster is flagged and removed to avoid other comparisons with other centroids in order that it cannot belong to another cluster.

Once this operation is completed, the first not flagged potential cluster centroid at the top of the list becomes the new cluster centroid c_1 and the same process of allocation is repeated until all potential cluster centroids are analyzed.

After this procedure of allocation, there is the possibility that there are some molecules that either not belong to the list of cluster centroids or are not related to any cluster, so these elements become singletons.

This can happen because some molecules identified as singletons may have neighbors given a Tanimoto similarity threshold, but those elements have been already associated to a cluster centroid with more weight in terms of number of neighbors.

Operating in this way, the resulting clusters are separated from each other and they are highly homogeneous. [36] [37]

Summarizing, the algorithm has these following features:

- creation of good quality clusters where the similarity within a cluster reflects the Tanimoto level used as threshold for the clustering
- using a high Tanimoto value leads to a very reliable method of grouping chemically similar molecules
- the more similar molecules ate in a giver cluster, the higher within similarity will be

4.4 Application of Butina algorithm

Butina algorithm is generally preferred mostly because the similarity within each cluster reflects al least the Tanimoto value used as a threshold while the clustering solutions of J-P algorithm strongly depend on the choices of its two parameters.

For this reason, Butina algorithm has been applied to our data using the similarity matrix based on Tanimoto distance to see how many homogeneous clusters are created by changing the threshold values regardless the 6 clusters identified based on *Activity* values.

We already have the similarity matrix between molecules, then the first two steps of the algorithm regarding fingerprints and calculation of similarity values are skipped.

The following table (*Table 1*) shows the results obtained by changing the Tanimoto threshold value between 0.2 and 0.9:

	N° clusters	**N° singletons**	**% molecules into clusters**	**Global similarity**
0.2	1	0	100%	0.349
0.3	10	2	99.88%	0.467
0.4	40	1	99.94%	0.537
0.5	53	5	99.71%	0.587
0.6	98	31	98.18%	0.661
0.7	276	445	73.88%	0.73
0.8	53	1548	9.15%	0.814
0.9	0	1704	0%	0

Table 1: results of Butina algorithm varying the Tanimoto threshold

More in detail, the table shows for each threshold value:

- number of clusters that are created (groups with more than one molecule inside)
- number of singletons (molecules that don't belong to any cluster)
- percentage of molecules the belong to any cluster
- average similarity considering all the clusters formed by the algorithm

Using 0.2 as threshold value, the algorithm returns a single cluster containing all 1704 molecules while the opposite situation happens by setting the parameter as 0.9 and for these reasons these 2 values must obviously be discarded.

Looking at the other cases, the value of average similarity reflects more or less the one used as a threshold.

Increasing the threshold value, the percentage of molecules present in any cluster decreases, especially going from 0.7 to 0.8.

In fact, using 0.8 as threshold value, we obtain 53 small clusters (smaller than 276 clusters using 0.7) and a lot of singletons.

To make a comparison with the 6 classes defined by the pharmaceutical company, setting 0.3 as threshold value, the algorithms produces 10 clusters with only 2 singletons and 99.88% of molecules considered for the formulation of these 10 clusters.

In the following table (*Table 2*), for each of the 10 clusters is reported its size and its within similarity value.

Cluster:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
N° mol.	1519	75	14	32	24	6	3	5	21	3
Sim.	0.38	0.39	0.49	0.48	0.44	0.57	0.56	0.45	0.5	0.4

Table 2: size and within similarity of each cluster using 0.3 as threshold

As we can easily see, the majority of molecules is focused on a single cluster, while others are quite small.

Bibliography

[1] M. Borrotti, D. De March, D. Slanzi, I. Poli, "*Designing Lead Optimisation of MMP-12 Inhibitors*", Hindawi Publishing Corporation, 2013

[2] Modulazione negativa
www.treccani.it/enciclopedia/modulazione-negativa_(Enciclopedia_della_ Scienza_e_ della_Tecnica)/

[3] Protein Function
http://www.nature.com/scitable/topicpage/protein-function-14123348

[4] Hit to lead
http://en.wikipedia.org/wiki/Hit_to_lead#Lead_optimization_phase

[5] S. Pickett, D. Green, D. Hunt, D. Pardoe, I. Hughes,
"*Automated Lead Optimization of MMP-12 Inhibitors Using a Genetic Algorithm*",
ACS Medicinal Chemistry Letters, 2011

[6] Structure-activity relationship
http://en.wikipedia.org/wiki/Structure–activity_relationship

[7] Quantitative structure-activity relationship
http://en.wikipedia.org/wiki/Quantitative_structure–activity_relationship

[8] Box-and-Whisker Plots: Interquantile Ranges and Outliers
http://www.purplemath.com/modules/boxwhisk3.htm

[9] D. Sarkar, "*Lattice: Multivariate Data Visualization with R.*", Springer, 2008

[10] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie,
"*The Elements of Statistical Learning: Data Mining, Inference, and Prediction*", Springer

[11] F. Giummolè, *Statistical Learning - Notes*

[12] E. Mooi, M. Sarstedt, "*A Concise Guide to Market Research*", Springer, 2011

[13] S. Terzi, "*La cluster analysis*"
http://mining2007.awardspace.com/eleonora/clusa.htm

[14] M. Dell'Omodarme, "*Esercitazioni di statistica biomedica: alcune note su R*"
http://cran.r-project.org/doc/contrib/DellOmodarme-esercitazioni-R.pdf

[15] Molecular fingerprints and similarity searching
http://openbabel.org/docs/dev/Fingerprints/intro.html

[16] D. Butina, "*Use of Chemical Similarity in Drug Discovery*"
http://mcc.irb.hr/mcc_04/presentations/butina_d_mcc04_2.pdf

[17] E. Rasmussen, "*Information Retrieval - Chapter 16: Clustering Algorithms*"
http://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/chap16.htm

[18] L. Kaufman, P. Rousseeuw, "*Finding Groups In Data: An Introduction to Cluster Analysis*", John Wiley and Sons Ltd, 2005

[19] G. F. von Borries, "*Partition clustering of High Dimensional Low Sample Size data based on P-values*"
http://krex.k-state.edu/dspace/bitstream/handle/2097/590/ GeorgevonBorries2008 .pdf?sequence=1

[20] S. Holland, "*Cluster Analysis*"
http://www.ioz.pwr.wroc.pl/Pracownicy/Lawik/res/Lab2-Holland_Cluster_ Analysis.pdf , 2006

[21] S. Orlando, "*Clustering*"
http://www.dais.unive.it/~dm/New_Slides/7_Cluster_DWM.pdf

[22] *Divisive Analysis (Diana)*
http://www.unesco.org/webworld/idams/advguide/Chapt7_1_5.htm

[23] F. Di Lascio, "*Analisi dei Gruppi con R*"
http://www2.stat.unibo.it/pillati/_doc/An_Gruppi.pdf

[24] Linux Kernel 2.6.29 + tux3
http://cs.jhu.edu/~razvanm/fs-expedition/tux3.html

[25] *Cluster Analysis*
http://www.stat.berkeley.edu/classes/s133/Cluster2a.html

[26] A. Struyf, M. Hubert, P. Rousseuw, "*Clustering in an Object-Oriented Environment*"
https://www.stat.washington.edu/wxs/Stat592-w2011/Literature/struyf-hubert-rousseeuw-clustering-object-oriented-1997.pdf

[27] D. Maringer, "*Cluster Analysis: Applied Data Analysis*"
http://wwz.unibas.ch/fileadmin/wwz/redaktion/qm/downloads/Lehre/ApplDA/ 11HS/ADA_11H_04_ClusterAnalysis.pdf

[28] P. Lewis, "*R for Medicine and Biology*", Jones and Bartlett Series in Biomedical Informatics, 2009

[29] M. Zaki, W. Meira Jr., "Data Mining and Analysis: Foundations and Algorithms"
http://www2.dcc.ufmg.br/livros/miningalgorithms/files/pdf/fdmaold.pdf

[30] I. Witten, E. Frank, "*Data Mining: Practical Machine Learning Tools and Techniques*", Morgan Kaufmann Publishers, 2011

[31] S. Ayramo, T. Karkkainen, "*Introduction to partitioning-based clustering methods with a robust example", 2006 http://users.jyu.fi/~samiayr/pdf/introtoclustering_report.pdf.*, 2006

[32] Package 'cluster'
http://cran.r-project.org/web/packages/cluster/cluster.pdf

[33] R. Jarvis, E. Patrick, "*Clustering Using a Similarity Measure Based on Shared Near Neighbors",* IEEE transactions on computer, 1973
http://davide.eynard.it/teaching/2012_PAMI/JP.pdf

[34] Improved Outcomes Software, "*Jarvis-Patrick Clustering Overview"*
http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Jarvis-Patrick_Clustering_Overview.htm

[35] B. Luke, "*Jarvis-Patrick Clustering"*
http://www.btluke.com/jpclust.html

[36] D. Butina, "*Unsupervised Data Base Clustering Based on Daylight's Fingerprint and Tanimoto Similarity: A Fast and Automated Way To Cluster Small and Large Data Sets*", J. Chem. Inf. Comput. Sci., 1999

[37] N. O'Boyle, *Taylor-Butina Clustering*
http://www.redbrick.dcu.ie/~noel/R_clustering.html