Université Paris Diderot - Paris 7
Master 1 Biologie - Informatique

Enoncé du TP long: Prediction of protein pocket druggability

Multivariate data analysis

Dr. Leslie Regad ; Dr. Anne Badel

Chapter 1

Presentation of the project

1.1 Aim of this project

Despite advances in both experimental and computational fields, around 60% of drug discovery projects fail because the target is not druggable (Brown et al., 2003). The druggability is the ability of a protein to bind a drug-like molecule, which corresponds to a large number of drugs present in the current pharmacopoeia. The drug-like molecules, defined for the first time in 1997 with Lipinski by "rules of five",3 have ADME properties (Absorption, Distribution, Metabolism, Excretion) specific allowing them to be absorbed and access in the body. Detecting the druggability of a target is a very important challenge in the discovery of new therapeutic targets in order to predict new binding sites of drug-like molecules.

The aim of this project is to identify the specific properties of druggable pockets and to develop a statistical model to predict the pocket druggability based on these properties. To perform that, you will test different machine learning methods such as CART, random forest, LDA, logistic regression on the data set composed of 109 protein pockets.

1.2 Presentation of data set

The dataset in the file `descriptor_pocket.mat` was extracted from the Krasowski's dataset and contains 109 pockets:

- 71 druggable pockets,
- 41 less druggable pockets.

Each pocket (druggable and undruggable) was described using a set of 28 descriptors computed using Fpocket (Le Guilloux et al., 2009) and python programs. These descriptors allow:

- quantification of the pocket geometry: `Real_volume, Mean_alpha.sphere_radius...`
- quantification of the amino acid composition of pockets (Pérot et al., 2013): frequencies of the 20 amino acids `A,C ...`

- the last descriptor `drugg` gives the druggability type of each pocket : 1 if the pocket is druggable, 0 if the pocket is less druggable.

Chapter 2

Analysis of the data

Remark: Before using a new R function(), see the help page of this function.

2.1 Preparation of the data

1. Open the file `descriptor_pocket.mat` containing the dataset.
2. Remove all pockets with missing values. Use the function `na.omit()`. How many pockets did you remove?

2.2 Computation of new variables

In this dataset, the composition of pockets is described using the frequency of 20 amino acids. However, it would be interesting to work with the amino acid classes:

- aromatic residues: F, Y, H, W
- polar residues: C, D,E, H, K, N, Q, R, S, T, W, Y
- aliphatic residues: I, L, V
- charged residues: D,E, R, K, H
- negative residues: D, E
- positive residues: H, K, R
- hydrophobic residues: C, G, A, T, V, L, I, M, F, W, Y, H, K
- small residues C, V, T, G, A, S, D, N, P
- tiny residues: A, C, G, S

1. Based on the class definition and the amino acid frequencies, compute the frequency of each class in the pocket dataset.
2. Add these new variables to the dataset. How many descriptors does you dataset contain?

2.3 Analysis of the dataset

2.3.1 Comparison of druggable and less druggable pockets

1. Which type of variables (quantitative, qualitative) does the dataset contain?
2. Plot the distribution of all descriptors using the function `boxplot()`. What do you observe?
3. Plot the distribution of all descriptors in druggable pockets and in less druggable pockets. What do you observe?
4. Compute the mean and the standard deviation values of each descriptor and in druggable and less druggable pockets. What do you observe?
5. Compare the mean values of each descriptor in druggable pockets and less druggable pockets. How many descriptors have the same mean in the 2 groups? Which are the descriptors that separate better druggable and less druggable pockets?

2.3.2 Correlation between descriptors

1. Compute the correlation between descriptors.
2. Represent the correlation matrix in a graphic. Which descriptors are the most correlated? What does it mean?
3. Plot (scatter-plot) the volume of each pocket (descriptor `Real_volume`) in function of its number of atoms (descriptor `C_atom`). Are these two descriptors significantly correlated?
4. Compute the linear regression to predict the pocket volume knowing its number of atoms.
5. Add on the scatterplot the regression line.
6. Analyze the models (residues, performances)
7. Predict the volume of the pockets containing 50, 200, 159 and 450 atoms.

All these steps allow obtaining a cleaned dataset. Now, you can begin the analysis of descriptors computed on druggable and less druggable pockets.

Chapter 3

Description of the pocket space and classification of pockets

functions: `scale()`, `dist()`, `hclust()`,`plot()`, `cutree()`, `table()`, `kmeans()`, `set.seed()`

3.1 Distances between pockets

To compare pockets is to compute a (dis)similarity measure between pockets. To estimate this measure, one solution is compute distances between pockets using pocket descriptors.

1. Give the definition of the Euclidean distance.
2. Compute the Euclidean distances between all pockets based on descriptors. Why do you use scale descriptor values?
3. Plot the distribution of distances between pockets. Which are the closest pockets? Are they extracted from the same protein family? Are they bind to the same ligand? Are they druggable or less druggable. To answer these questions use the web sites `http://www.rcsb.org`, `http://www.ebi.ac.uk/pdbsum/` and `http://www.uniprot.org/`.
4. Estimate the (dis)similarity measure between pockets using other distances. Compare the obtained results.

3.2 Classification of pockets

Now you will use 2 unsupervised methods to classify pockets according to their geometry and their physico-chemical properties.

3.2.1 Hierarchical classification

The first method that you will use is the hierarchical classification. This approach performs a hierarchical cluster analysis using a set of dissimilarities for the n objects being clustered.

Initially, each object is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster.

1. What are the input and output of the `hclust` function?

The hierarchical classification can be computed using different distance: Euclidean, Manhattan, Maximum, ... You will work with the Euclidean distance matrix computed on pockets using the un-correlated descriptors.

2. Compute the Euclidean distance based on the non-correlated descriptors between pockets.
3. Compute the hierarchical classification using average aggregation method.
4. Analysis of the computed hierarchical classification
   - Visualize the obtained dendrogram. Comment the tree structure.
   - Test the robustness of the dendrogram by testing different aggregation method. Is the dendrogram stable in terms of tree-structure.
   - Modify the dendogram labels to indicate the pocket druggability. What do you observe?
5. Extraction and analysis of pocket clusters

   To extract pocket clusters, you need to cut the obtained dendogram. The difficulty is to know how many clusters to extract. This question is a very large domain of research. One solution to select the optimal number of cluster is to study the quality of different partitions extracted from the same classification. To do so, we will use the computation of the data variability. The data variability of the data is splitting in:

   - intra-variability (I_W) that corresponds to the dispersion of point around the center of its cluster. This variability must be minimize
   - inter-variability (I_B) that corresponds to the distance between gravity centers of clusters to the gravity center of the dataset, noted g.

   Let a partition of K clusters with gravity centers $(g_1, g_2, ..., g_k)$. The total variability of the data (I_T) is decomposed in :

$$I_T = I_W + I_B \tag{3.1}$$

$$I_T = \sum_{k=1}^{K} \sum_{i \in k} d^2(i, g_k) + \sum_{k=1}^{K} d^2(g_k, g) \tag{3.2}$$

   The optimal partition corresponds to the partition that minimize I_W and maximize I_M . The quality of a partition can be estimated by :

   - I_W: the data variability explained by clusters

- R^2: the proportion of variability explained by clusters

$$R^2 = \frac{I_W}{I_T}$$

The 2 quality parameters may serve as a useful guideline for select the optimal cluster number. Different partitions are extracted from the classification. For each partition, these two quality parameters are computed and plotted in function of number of clusters of each partition. The optimal number of clusters can be chosen from these 2 graphics. It corresponds to the number of cluster where successive values of I_W and R^2 make a sudden jump.

- Compute the different partitions containing from 2 to 15 clusters from the obtained pocket classification
- For each partition, compute the occurrence of each cluster. Can you remove some partitions?
- For the remaining partition, compute the two quality parameters. Plot these values in function of the number of clusters? What is the optimal cluster number?
- Analyze the pocket clusters according to the descriptors. What is the specificity of each cluster?
- Analyze the clusters according to the druggability.

3.2.2 K-means

K-means clustering is a method to partition n observations into k clusters, in which each observation belongs to the cluster with the nearest mean. The disadvantage of K-mean method is that you must know the number of clusters.

1. What are the input and output of the `kmeans()` function?
2. To test the kmeans procedure, firstly compute a kmeans with 2 clusters using the pocket descriptors. What are the outputs of the function? Are the group is homogeneous in terms of druggability?
3. As the K-means seeds are randomly chosen, it is important to perform the K-means several times. Re-preformed the mean procedure. What do you observe? Compare the occurrence of obtained groups. Are the kmeans results stable? Think to modify the seed of the random procedure.

An important parameter in Kmeans procedure is the number of clusters to define. To determine this optimal number, you will use the procedure based on the analysis of the partition quality using parameters (R^2, I_W). This procedure is the same used in hierarchical classification. The `kmeans()` function computes directly the intra-variability of each cluster.

5. Using this procedure, determine the optimal cluster number for pocket dataset.

6. Extract clusters from a K-means method using this optimal number. Compare the obtained partition with one extracted from the hierarchical classification.

3.2.3 Mixed method

The mixed approach consists to combine the advantages of the hierarchical classification and kmeans approaches. It consists in:

- Step 1: Computation of an hierarchical classification from the dataset
- Step 2: Define the optimal number k of clusters from this classification using I_W and R^2 parameters of different partitions.
- Step 3: Computation of the k clusters and their centres of gravity.
- Step 4: Computation of a Kmeans procedure using the k centres of gravity as centers.

The advantages of this method are that the initial centers are not randomly choosen and some individual reassignments are allowed.

1. Classify pockets using the mixed approach.

2. Are these clusters homogeneous in terms of druggability?

3. Which descriptors do separate the different pocket clusters?

4. Compare the partitions obtained using a hierarchical classification and kmeans.

3.3 Description and analysis of the pocket space

libraries: `FactoMineR`
functions: `library()`, `PCA()`^{FactoMineR}, `sample()`.

As the dataset contains more than 2 variables, it is difficult to visualize dataset using classical plot. A way to overcome this problem is the use of multivariate analysis. As the descriptors correspond to quantitative variables, you will use a Principal Component Analysis (PCA). This method allows a reduction of the number of variables after converting the set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. To perform the PCA, you will use the library `FactoMineR`.

1. Import the library `FactoMineR`.

2. Do you compute the PCA using variance/co-variance matrix or the correlation matrix. Justify your answer.

3. Compute the PCA from pocket descriptors without the graph representation.

4. Analyze the obtained outputs.

5. To analyze the variability explained by each CP, plot the cumulate sum of data variability explained by the different PCs (= cumulate sum of eigen values).

6. Plot the projection of descriptors in the plane defined by the two first PCs (first PCA plane). This graph allows:

   - analyzing the correlation between variables
   - analyzing the first PCA space using descriptors.

   Analyze this graph: Which descriptors explain the first PC? Which descriptors explain the second PC? Are some descriptors correlated?

7. Plot the projection of pockets in the first PCA plane. What do you observe? Do you observe some outliers? Do you identify some pocket clusters? Explain the identified pocket clusters using variable information.

8. To analyze the druggability according to descriptors, project pockets onto the first PCA plane and color them according to their druggability. What do you observe? Explain druggable and less-druggable pockets in terms of descriptors.

9. To characterize pocket clusers obtained during the mixed procedure (hierarchical classification + kmeans), project pockets onto the PCA first plane and color them according to the obtained clusters. What do you observe? Explain these clusters using descriptors.

10. Analyze the second PCA plane composed by the third and fourth PCs using the same procedure.

Supplementary questions

The PCA analysis is often used as to reduce the dimensionality of data, and for producing insight as to structure of latent variables (cf. Velicer & Jackson, 1990). Instead of working with the original data matrix (n observations describing by p variables), we can work with the c most variable CPs. In this case, these c CPs are the new variables defined as the linear combinations of the p initial variables.
There are several approaches to determine how many CPs to retain:

- The Kaiser rule (1960) is the best know and most utilized in practice (Fabrigar et. al, 1999). According to this rule, only the CPs that have eigenvalues (variability) greater than one (defined threshold) are retained for interpretation or analysis.

- Scree test of Cattell (1966): Cattell's Scree test corresponds to a visual exploration of a graphical representation of the CP eigenvalues. In this method, the eigenvalues of each CP are in descending order. Afterwards, the graph is examined to determine the elbow point. The CPs before this elbow point are then conserved for future analyses.
- The parallel Horn's analysis (1965). This method is based on the generation of random variables, to determine the number of CPs to retain. This Parallel Analysis (PA) implies a Monte Carlo simulation process, since 'expected' eigenvalues are obtained by simulating normal random samples that parallel the observed data in terms of sample size and number of variables. A CP was considered significant if the associated eigenvalue was bigger than the mean of those obtained from the random uncorrelated data. Currently, it is recommended to use the eigenvalue that corresponds to a given percentile, such as the 95th of the distribution of eigenvalues derived from the random data (Cota et al., 1993; Glorfeld, 1995).

1. Using the parallel Horn's Analysis, compute how many CPs are significant using pocket datasets.
2. Compute a classification of pocket using the mixed approach from c selected CPs.
3. How many pocket clusters do you extract?
4. Analyze these clusters in terms of occurrence and descriptors.
5. Compare these clusters to those obtained using all variables.

3.3.1 Visualization of the pocket space using Multi Dimensional Scaling (MDS)

functions: `cmdscale()`, `dist()`.

We will use a second method, the Multi Dimensional Scaling (MDS), to visualize the pocket space. The MDS allows you to create a plot translating the pocket space, where two close pockets correspond pockets similar in terms of descriptors.

1. See the help page of the `cmdscale()` function. What type of input does the `cmdscale()` function?
2. Compute the MDS on the Euclidean distance matrix between pockets.
3. Plot the pocket projection into the two first axes computed by the MDS. Color pockets according to their druggability. What do you observe?
4. Compare the MDS and PCA results.
5. Color pockets projected into the MDS plane according to clusters obtained by the mixed procedure. What do you observe?

Chapter 4

Prediction of the pocket druggability

The aim of this part is to develop statistical models to predict the pocket druggability. It is a prediction of two classes: (i) druggable and (ii) undruggable. In this chapter, we will build druggability models using different machine-learning approaches: Linear Discriminant Analysis, Logistic Regression, CART, random Forest. The comparison of these druggability models allows us to identify the best models to predict the pocket druggability. Before building models, we must prepare dataset.

4.1 Preparation steps

functions: `sample()`

4.1.1 Training and test sets

To estimate the performance of statistic models, you need to create:

- a training set to train the model
- a test set to test and validate the computed model.

To create the training set, choose randomly 2/3 of pockets. The validation set will be composed of the remaining 1/3 of pockets. The repartition of druggable and less druggable pockets in these two sets must be close to their repartition in the initial data set.

1. Create a function to create a training and a test sets.
2. Apply this function to the pocket dataset.

4.1.2 Cross-validation

To quantify the robustness of a statistical model, you will use a cross-validation procedure. The k-fold cross-validation consists:

- to split of the training set in k groups

- train the model on the $k-1$ groups: training step
- test the model on the remaining group: test step

This procedure is run until all groups are used in the test step.

1. Create a function that generate the k groups from the dataset.

4.1.3 Parameters to quantify the performance of a statistical model

There are different parameters to measure the performance of a statistical model. They are based on the computation of the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (NF), see Table 4.1:

- **Accuracy** that corresponds to the proportion of well-predicted pockets.

$$\frac{TP+TN}{TP+TN+FP+FN}$$

- **MCC** (Matthew's Correlation Coefficient). This parameter takes into account the unbalance between the number of druggable and less druggable pockets in the dataset.

$$MCC = \frac{TP \times TN - FP \times FN}{(TP+FP) \times (TP+FN) + (TN+FP) \times (TN+FN)}$$

- **Sensitivity** or recall is the probability to predict druggable pockets.

$$\frac{TP}{TP+FN}$$

- **Specificity** is the probability to predict less druggable pockets.

$$\frac{TN}{TN+FP}$$

- **Precision** is the proportion of real druggable pockets amongst the predicted druggable pockets.

$$\frac{TP}{TP+FP}$$

		True values	
		druggable	less druggable
Predicted values	druggable	TP	FP
	less druggable	FN	TN

Table 4.1: Computation of counts to quantify the performance of a statistical model. TP: true positives, TN: true negatives, FP: false positives, FN: false negatives.

1. Create functions that compute these parameters

4.2 Linear Discriminant Analysis (LDA)

library: `MASS`
functions: `lda()`^{MASS}, `predict()`

LDA is a machine learning method to find linear combinations of features, which characterizes or separates two or more classes of objects or events. The resulting combinations may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

1. Justify the LDA model use to compute a druggability model.
2. Compute a complete LDA model ((model with all variables) to predict pocket druggability using the training set.
3. Compute the different parameters quantifying the quality of the model (accuracy, sensitivity, specificity, ...) on training set.
4. Test the model using the test set.
5. Test the robustness of the model by computing the model performence using a 4-fold cross-validation
6. Sum up the model performance in a table.
7. Plot the score associated to each variable in the model. Which variables are the most involved in the model?
8. Compute a new LDA-model using only these important variables? Compute the performances of this model on training and test sets, and in 4-fold cross-validation.
9. Compare the performance of the two LDA-models.

4.3 Logistic regression

functions: `glm()`, `predict()`, `step()`

1. Justify the logistic model use to compute a druggability model.
2. Compute a logistic model using all descriptors on the training set.
3. Compute the performance of this model using the training set and a probability threshold of 0.5.

4. Compute the performance of this model using the test set and a 4-fold cross-validation

5. Which are the most significant variables in this model?

An advantage of the logistic regression is that it performs variable selection. This selection is based on the Akaike information criterion (AIC), which is a measure of the relative goodness of fit of a statistical model.

$$AIC = 2k - 2ln(L)$$

where k is the number of parameters in the statistical model, and L is the maximized value of the likelihood function for the estimated model. Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value.

6. see the help page of `step()` function.

7. Starting with the complete model select the most significant descriptors using the `both` procedure.

8. Compute the performance of this model using the training and test sets and 4-fold cross-validation and a probability threshold of 0.5.

9. Analyze the evolution of the AIC parameter during the model selection step.

10. Which are the selected descriptors?

Classically, the variable selection step is performed with a cross-validation procedure. Here, we will re-run the variable selection using a 4-fold cross-validation using 20 iterations.

11. During each fold of a cross-validation, conserve the selected descriptors. At the end of each 4-fold cross-validation procedure (20 iterations), you have four groups of conserved descriptors. After the 20 iterations, compute the occurrence of each variable in all models. Which descriptors are the most used? Is there a stability in the descriptor selection?

12. Compute a new model using these selected descriptors and its performance on training and test sets.

13. Evaluate the robustness of the model using a 4-fold cross-validation

14. Store the performance of all logistic models in a table.

4.4 Classification And Regression Tree (CART)

library: `rpart`
functions: `rpart()`^{rpart}, `prune()`, `plotcp()`, , `printcp()`

The CART methodology is technically known as binary recursive partitioning. The process is binary because parent nodes are always split into exactly two child nodes and recursive because the process can be repeated by treating each child node as a parent. The key elements of a CART analysis are a set of rules for:

- splitting each node in a tree;
- deciding when a tree is complete;
- and assigning each terminal node to a class outcome (or predicted value for regression).

1. Compute a CART model on the training set. Extract all variables implied in the tree.
2. Plot the obtained tree.
3. Compute the performance of this model using the training set. The more node the tree have, the best its performances in resubstitution are and the worse its performances in generalization are.
4. Compute the performance of this model using the test set.

You must decrease the number of leaves by pruning the tree using the approach based on the complexity parameter. Typically, one selects a tree size that minimizes the cross-validated error, the xerror column printed by the function `printcp()`. The cross-validation results can be visualized using the function `plotcp()`: it plots the decrease of the relative error (error divided by the variance of the predictive variable) in function of the complexity parameter. This representation gives a good choice of complexity for pruning: leftmost value for which the mean lies below the horizontal line.

4. Select the complexity value to prune the tree using `printcp()` function.
5. Prune the tree using the function `prune()` and plot the pruned tree.
6. Compare the variable involved in the pruned and complete trees.
7. Compute the performance of the pruned tree using the training and test sets and a 4-fold cross-validation.
8. Sum up the model performance on a table.

4.5 Random Forest

library: `randomForest`
functions: `randomForest()`^{randomForest}, `prune()`, `plotcp()`, , `printcp()`

To compute a random forest (RF) model, you will use the function `randomForest()` available in the library `randomForest`.

4.5.1 Step 1: Determination of the optimized parameters of RF model

The first step of the computation of RF model is the determination of optimized parameters:

- *ntree*: Number of trees to grow.
- *mtry*: Number of variables randomly sampled as candidates at each split.

To optimize these parameters, we will use a grid approach by testing different values of *ntree* and *mtry* values: $ntree = \{10, 50, 100, 200\}$ and $mtry = \{5, 10, 20, p\}$.

1. Compute the accuracy value of each model using the differents (*ntree*, *mtree*) pairs and fill in Table 4.2 with the accuracy values.
2. Determine the optimized value of *ntree* and *mtree* defined as the pair (*ntree*, *mtry*) maximizing the accuracy during the grid approach.
3. Visualize the obtained matrix. Which (*ntree*, *mtree*) pairs do you retain?
4. Compute performances of the model using the optimal *ntree* and *mtree* parameters on training and test sets and 4-fold cross-validation.

		ntree				
		10	50	100	150	200
mtry	5					
	10					
	20					
	p					

Table 4.2: Grid of *ntree* and *mtry* pairs

4.5.2 Step 2: Selection of the most informative variables

As CART method, RF method allows the selection of most informative variables. This selection is based on the computation of the "Importance" of each variable. RF variable importance, noted $I(X^j)$ of variable X^j is defined as follows (Genuera R. *et al.*, *Pattern Recognition Letters* 2010). For each tree t of the forest, consider the associated OOB_t sample (data not included in the boostrap sample used to construct t). Denote by $errOOB_t$ the error (misclassification rate for classification) of a single tree t on this OOB_t sample. Now, randomly permute the values of X_j in OOB_t to get a perturbed sample denoted by OOB_t^{j*} and compute $errOOB_t^{j*}$, the error of predictor t on the perturbed sample. Variable importance of X^j is then equal to:

$$I(X^j) = \frac{1}{ntree} \sum_t (errOOB_t^{j*} - errOOB_t)$$

where the sum is over all trees t of the RF using the variable X^j.

1. Using the optimized parameters compute a RF on the training set and extract the important variables. Store these values of variable importance in a vector.
2. As RF method is based on random, re-run this procedure 100 times.
3. Plot the average importance for all variables. Select the most informative variables.
4. Re-compute a RF model, using the optimal parameters and the most informative variables. Compute the performance parameters on the training and test sets and using a 4-fold cross-validation.

4.6 Comparison of different models and selection of the best model to predict pocket druggability

1. Compare the performance parameters of all build models. Which model does allow the best separation of druggable and less druggable pockets?
2. Compare the significant descriptors for each model.

A second method to compare different models is the plot of the ROC (Receiver Operating Characteristic) curve. This curve is a graphical plot, which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the sensitivity of the model *versus* (1-specificity) at various threshold settings. The best model corresponds to the model with the highest Area Under the Curve (AUC).

1. For models obtained by LDA, CART, RF and logistic regression compute the sensitivity and specificity values for a probability threshold varying between 0 and 1. Be careful, the function `predict()` must return the probability to belong to a class.
2. Plot the ROC curve for these models. Which model allows the best separating between the druggable and less druggable pockets?

Acknowledgments

- Thanks to Alexandre Borrel to the extraction of the dataset and the computation of the descriptors.
- Thanks to Olivier Taboureau, Alexandre Borrel and Anne Badel for the reviewing of the manuscript.