Analyse de Données

Outils : Logiciel R, Environnement de Développement Intégré RStudio

Nous allons utiliser durant cette étude de cas les techniques d'extraction de règles d'association afin d'identifier des liens entre valeurs de variables dans les données et les techniques de clustering afin d'identifier des clusters (groupes) d'instances similaires, i.e. définies par des valeurs de variables proches.

Rappel : la liste des instructions exécutées lors de la précédente session est stockée dans le fichier `.Rhistory` automatiquement créé dans le répertoire de travail lorsque vous quittez l'environnement R. Les instructions `help(cmd)` et `? cmd` permettent d'afficher l'aide sur la commande `cmd`.

1 Itemsets Fréquents et Règles d'Associations

Les techniques d'analyse de liens entre valeurs des variables seront mises en œuvre sur des données de transactions de ventes correspondant à des paniers d'achats dans le commerce de détail.

L'objectif final de cette application est d'identifier les comportements d'achats fréquents parmi les clients, c'est-à-dire les articles fréquemment achetés ensembles

1.1 Ensemble de données *Data Groceries*

L'ensemble de données *Data Groceries* contient 9 836 tickets de transactions de ventes de détail. Cet ensemble représenté sous forme d'une matrice binaire, décrit pour chaque transaction la liste des articles achetés ensemble.

- Chargez dans R l'ensemble de données `Data Groceries.csv` dans un data frame `groceries` à l'aide de la fonction `read.table()` par la commande :

```
> groceries <- read.table("IA360 - Module 4 - Analyse de Données - Data
  Groceries.csv", header=TRUE, dec=".", sep="\t", stringsAsFactors=T)
```

- Affichez le nombre total d'items (valeurs des variables binaires) à l'aide de la fonction `ncol()` par la commande :

```
> ncol(groceries)
```

- Affichez la liste des items (colonnes) à l'aide de la fonction `colnames()` par la commande :

```
> colnames(groceries)
```

1.2 Exploration des données

- Installez la librairie *arules* dans R par la commande :

```
> install.packages("arules")
```

- Activez la librairie *arules* dans votre session R par la commande :

```
> library("arules")
```

Nous allons commencer par examiner quels sont les items (articles) les plus fréquents. Pour cela, la fonction `itemFrequencyPlot()` requiert un ensemble de données au *format transactionnel*.

- Générez un ensemble de données au format transactionnel `groceries_tr` à partir du data frame `groceries` à l'aide de la fonction `as()` par la commande :

```
> groceries_tr <- as(groceries, "transactions")
```

- Affichez les informations résumées de l'ensemble de données `groceries_tr` à l'aide de la fonction `summary()` par la commande :

```
> summary(groceries_tr)
```

- Affichez l'histogramme d'effectifs des 20 items les plus fréquents par la commande :

```
> itemFrequencyPlot(groceries_tr, topN=20)
```

  Le support de chaque item est représenté sur l'axe des ordonnées en valeur relative (pourcentage d'instances contenant l'item).

- Affichez l'histogramme d'effectifs avec le support des items en valeur absolue pour les 20 items les plus fréquents par la commande :

```
> itemFrequencyPlot(groceries_tr, topN=20, type="absolute")
```

Le support de chaque item est représenté sur l'axe des ordonnées en valeur absolue (nombre de lignes d'occurrence de l'item).

☛ Affichez l'histogramme d'effectifs avec le support des items en valeur absolue (nombre d'instances) pour les items de support ≥ 10 % par la commande :

```
> itemFrequencyPlot(groceries_tr, support=0.10)
```

☛ Fixez à 3 le nombre de décimales affichées pour les mesures (support, confiance, etc.) associées aux itemsets et règles d'association par la commande :

```
> options(digits=3)
```

1.3 Extraction d'itemsets fréquents

☛ Affichez le nombre total d'itemsets potentiellement fréquents (taille de l'espace de recherche des itemsets fréquents, appelé *treillis des itemsets*) par la commande :

```
> 2^ncol(groceries_tr)
```

Nous allons utiliser la fonction `apriori()` pour extraite les itemsets fréquents et règles d'association.

☛ Extrayez les itemsets fréquents pour un seuil minimal de support de 2% à l'aide de la fonction `apriori()` et stockez le résultat dans un objet `fi` par la commande :

```
> fi <- apriori(groceries, parameter = list(supp = 0.02, target = "frequent
  itemsets"))
```

☛ Affichez les itemsets fréquents stockés dans l'objet `fi` par la commande :

```
> inspect(fi)
```

☛ Affichez un histogramme des tailles des itemsets fréquents dans `fi` par la commande :

```
> barplot(table(size(fi)), xlab="Taille itemset", ylab="Nombre")
```

☛ Triez l'ensemble `fi` par ordre décroissant des supports des itemsets à l'aide de la fonction `sort()` de la librairie *arules* par la commande :

```
> fi <- sort(fi, by="support")
```

☛ Affichez la liste des 30 premiers itemsets fréquents avec leur support dans l'ensemble `fi` par la commande :

```
> inspect(head(fi, n=30))
```

Note : la commande `inspect(fi[1:30,])` permet d'obtenir le même résultat.

☛ Affichez la liste des itemsets fréquents de taille ≥ 3, i.e. constitués de plus de 2 items, par la commande :

```
> inspect(fi[size(fi)>2])
```

1.4 Extraction de Règles d'association

Nous souhaitons extraire les règles d'association les plus pertinentes montrant les liens entre les achats de deux articles.

☛ Extrayez les règles d'association pour un seuil minimal de support de 1% et un seuil minimal de confiance de 40% dans un objet `rules1` à partir de l'ensemble de données `groceries` par la commande :

```
> rules1 <- apriori(groceries, parameter = list(supp = 0.01, conf = 0.4,
  target = "rules")
```

☛ Affichez les règles d'association extraites dans l'objet `rules1` par la commande :

```
> inspect(rules1)
```

Les règles d'association extraites sont de la forme :

```
    lhs                  rhs               support confidence coverage lift count
[1] {Hard.cheese=Y}  =>  {Whole.milk=Y}    0.0101  0.411      0.0245   1.61  99
```

dans laquelle :

- `lhs` est l'itemset antécédent de la règle,
- `rhs` est l'itemset conséquence de la règle,
- `support` est la proportion d'instances contenant `lhs` et `rhs`,
- `confidence` est la proportion d'instances contenant `rhs` parmi celles contenant `lhs`,
- `coverage` est la proportion d'instances contenant `lhs`,

- `lift` mesure la corrélation statistique entre `lhs` et `rhs` :

  `lift(lhs → rhs) = P(lhs ∪ rhs) / (P(lhs) . P(rhs)).`

Note : la mesure de corrélation statistique du *lift* permet de filtrer des règles non significatives potentiellement générées si certains items ont une fréquence très élevée dans les données. Un *lift* = 1 indique l'indépendance entre les occurrences de `lhs` et `rhs`, un *lift* < 1 indique une dépendance négative entre `lhs` et `rhs`, et un *lift* > 1 indique une dépendance positive entre `lhs` et `rhs`. Les règles de *lift* ≤ 1 seront donc ignorées.

☛ Triez les règles extraites dans `rules1` par ordre décroissant de confiance et de support respectivement par la commande :

```
> rules1 <- sort(rules1, by = c("confidence", "support"))
```

☛ Réaffichez les règles dans `rules1` à l'aide de la fonction `inspect()`.

Le paramètre `parameter` de la fonction `apriori()` permet de définir le format (taille minimale et maximale, etc.) des règles extraites.

☛ Extrayez les règles d'association d'une taille maximale de deux items, i.e. un item en antécédent et un item en conséquence, pour un seuil minimal de support de 1% et un seuil minimal de confiance de 40% dans un objet `rules2` par la commande :

```
> rules2 <- apriori(groceries, parameter = list(supp = 0.01, conf = 0.4, target =
  "rules", maxlen=2))
```

☛ Affichez les règles dans `rules2` à l'aide de la fonction `inspect()`.

☛ Extrayez les règles d'association d'une taille maximale de deux items pour un seuil minimal de support de 0,1% et un seuil minimal de confiance de 50% dans un objet `rules3`.

☛ Triez les règles dans `rules3` par ordre décroissant de confiance et de support respectivement à l'aide de la fonction `sort()`.

☛ Affichez les règles dans `rules3` à l'aide de la fonction `inspect()`.

1.5 Ciblage d'items

Le paramètre `appearance` de la fonction `apriori()` permet de définir quel(s) item(s) doivent apparaître dans l'antécédent et/ou la conséquence des règles extraites.

☛ Extrayez les règles d'association contenant l'item `Whole.milk=1` en antécédent (paramètre `lhs`) pour des seuils minimaux de support et confiance de 2% et 20% respectivement par la commande :

```
> wm1 <- apriori(data=groceries, parameter=list(supp=0.01, conf=0.2),
  appearance=list(lhs="Whole.milk=Y"), control=list(verbose=F))
```

☛ Triez les règles extraites dans l'objet `wm1` par ordre décroissant de confiance et de support respectivement à l'aide de la fonction `sort()` et affichez les à l'aide de la fonction `inspect()`.

☛ Extrayez les règles d'association contenant l'item `Whole.milk=Y` en conséquence (paramètre `rhs`) pour des seuils minimaux de support et confiance de 2% et 55% respectivement par la commande :

```
> wm2 <- apriori(data=groceries, parameter=list(supp=0.01, conf=0.55),
  appearance=list(rhs="Whole.milk=Y"), control=list(verbose=F)))
```

☛ Triez les règles extraites dans l'objet `wm2` par ordre décroissant de confiance et de support respectivement à l'aide de la fonction `sort()` et affichez les à l'aide de la fonction `inspect()`.

1.6 Représentations graphiques des règles

Nous allons utiliser la librairie *arulesViz* afin d'obtenir des représentations graphiques des règles.

☛ Installez et activez la librairie *arulesViz* pour la visualisation graphique de règles d'association.

☛ Affichez les règles de l'ensemble `wm1` sous forme de graphe par la commande :

```
> plot(wm1, method="graph", shading="confidence")
```

Le paramètre `shading` permet de sélectionner la mesure (e.g. support, confiance, lift) utilisée pour définir l'intensité de la coloration des nœuds : plus la couleur est intense, plus la mesure est élevée.

☛ Affichez les règles de l'ensemble `wm2` sous forme de graphe interactif par la commande :

```
> plot(wm2, method="graph", shading="confidence", engine='interactive')
```

L'interactivité du graphique permet de déplacer à l'aide de la souris les nœuds du graphe affiché afin d'améliorer sa lisibilité ; les menus proposent des organisations prédéfinies (aléatoire, en cercle, etc.).

☛ Affichez les règles de l'ensemble `wm2` sous forme de boulier interactif par la commande :

```
> plot(wm2, method="grouped", shading="confidence", engine='interactive')
```

L'interactivité du graphique permet de sélectionner un disque à l'aide de la souris et afficher textuellement la règle correspondante en cliquant sur le bouton `inspect`.

☛ Affichez les règles de l'ensemble `rules3` sous forme de matrice par la commande :

```
> plot(rules3, method="matrix", shading="confidence", engine='interactive')
```

Cette représentation permet d'explorer de grands ensembles de règles de façon simplifiée. Cliquer sur l'une des cellules de la matrice permet d'afficher textuellement la règle correspondante.

2 Clustering de Données

L'ensemble de données *Data Iris.csv* décrit les caractéristiques de 150 fleurs appartenant à trois espèces d'iris. Pour chaque fleur, les longueurs et largeurs des sépales et pétales sont répertoriées ainsi que l'espèce d'iris correspondante (*Iris-setosa*, *Iris-versicolor* ou *Iris-virginica*).

Caractéristiques des données :

- Instances : 150 fleurs
- Nombre de variables : 5
- Variable de classe : Species
- Valeurs manquantes : aucune
- Séparateur de colonnes : virgule
- Séparateur de décimales : point

Description des Variables

Variable	Type	Description	Domaine de valeurs
Sepal.Length	Réel	Longueur des sépales	[4.3, 7.9]
Sepal.Width	Réel	Largeur des sépales	[2.0, 4.4]
Petal.Length	Réel	Longueur des pétales	[1.0, 6.9]
Petal.Width	Réel	Largeur des pétales	[0.1, 2.5]
Species	Texte	Espèce de la fleur	Iris-setosa, Iris-versicolor, Iris-virginica

☛ Chargez les données du fichier *Data Iris.csv* dans un data frame `iris` à l'aide de la fonction `read.csv()` par la commande :

```
> iris <- read.csv("IA360 - Module 4 - Analyse de Données - Data Iris.csv",
  header = TRUE, sep = ",", dec = ".", stringsAsFactors = T)
```

2.1 Visualisation bidimensionnelle des données

Nous allons utiliser la librairie *ggplot2* qui fournit des outils avancés pour l'affichage de graphiques afin d'afficher les données en deux dimensions (longueur et largeur) pour les pétales et les sépales des fleurs.

Nous allons pour cela afficher un nuage de points pour les pétales et un nuage de points pour les sépales dans lesquels la couleur de chaque point correspondra à l'espèce de la fleur (variable `Species`).

☛ Installez et activez la librairie *ggplot2*.

☛ Affichez un nuage de points des longueur (`Length`) et largeur (`Width`) des sépales de chaque fleur avec l'espèce en couleur par la commande :

```
> qplot(Sepal.Length, Sepal.Width, data=iris, color = Species)
```

☛ Affichez un nuage de points des longueur (`Length`) et largeur (`Width`) des pétales de chaque fleur avec l'espèce en couleur par la commande :

```
> qplot(Petal.Length, Petal.Width, data=iris, color = Species)
```

2.2 Clustering des données par algorithme des K-means

La fonction `kmeans()` permet de réaliser un clustering par la méthode des *K-means* à partir d'un data frame contenant des variables numériques ou bien d'une matrice de distance.

☛ L'initialisation des centroïdes, qui est la première étape de la méthode des *K-means*, étant aléatoire,

nous allons définir la suite aléatoire qui sera utilisée afin de s'assurer de la reproductibilité des résultats par la commande `set.seed()` :

```
> set.seed(1234)
```

☛ Exécutez le clustering par *k-means* du data frame `iris` en :

- utilisant les colonnes 1 à 4 comme variables en entrée,
- fixant le nombre de clusters générés à 3,
- exécutant 20 initialisations aléatoires afin de sélectionner celle donnant la plus faible variance intra-clusters,

et en stockant le résultat dans un objet nommé `km3` par la commande :

```
> km3 <- kmeans(iris[, 1:4], 3, nstart = 20)
```

☛ Ajoutez le numéro de cluster d'affectation de chaque instance sous la forme d'une nouvelle colonne nommé `Cluster` dans le data frame `iris` par la commande :

```
> iris$Cluster <- as.factor(km3$cluster)
```

☛ Affichez la liste des numéros (1 à 3) de clusters d'affection pour chacune des 150 instances par la commande :

```
> print(iris$Cluster)
```

☛ Affichez un nuage de points des longueur et largeur des sépales avec les clusters d'affectation des instances représentés par des symboles différents par la commande :

```
> qplot(Sepal.Length, Sepal.Width, data = iris, pch = iris$Cluster) +
  geom_point(size = 3)
```

La forme de chaque point indique le numéro du cluster d'affectation de l'instance représentée par le point.

☛ Affichez un nuage de points des longueur et largeur des pétales avec les clusters d'affectation des instances représentés par des symboles différents par la commande :

```
> qplot(Petal.Length, Petal.Width, data = iris, pch = iris$Cluster) +
  geom_point(size = 3)
```

2.3 Évaluation externe des clusters selon l'espèces des iris

Nous allons maintenant évaluer dans quelle mesure chacun des trois clusters générés permet de distinguer une espèce d'iris particulière.

Afin d'évaluer la correspondance entre les regroupements des instances (clusters) et la classe des instances (espèce), nous allons comparer les proportions de chaque espèce pour chaque cluster.

☛ Affichez une table de contingence affichant pour chaque cluster (en ligne) le nombre d'instances de chaque espèce (en colonne) par la commande :

```
> table(iris$Cluster, iris$Species)
```

☛ Affichez un histogramme des effectifs des clusters avec les classes en couleurs par la commande :

```
> qplot(iris$Cluster, data=iris, fill=iris$Species)
```

Chacune des 3 barres affichées correspond à un cluster et la proportion de chacune des trois couleurs représente les instances de chacune des trois espèces parmi les instances des clusters. Une barre d'une unique couleur représente ainsi un cluster dont toutes les instances sont de la même espèce.

Nous allons maintenant comparer graphiquement la correspondance entre le cluster d'affectation et l'espèce de chacune des 150 fleurs.

☛ Affichez un nuage de points des longueur et largeur des sépales avec l'espèce de l'instance représentée par la couleur du point et le cluster d'affection de cette instance représenté par son symbole par la commande :

```
> qplot(Sepal.Length, Sepal.Width, data=iris, color=Species, pch=iris$Cluster) +
  geom_point(size = 3)
```

☛ Affichez un nuage de points des longueur et largeur des pétales avec l'espèce de l'instance représentée par la couleur du point et le cluster d'affection de cette instance représenté par son symbole par la commande :

```
> qplot(Petal.Length, Petal.Width, data=iris, color=Species, pch=iris$Cluster) +
  geom_point(size = 3)
```