Analyse des données — Master 2 IMAFA
TD 2—transformation et analyse

Andrea G. B. Tettamanzi
Université Nice Sophia Antipolis
andrea.tettamanzi@unice.fr

Vendredi 19 janvier 2018

Résumé

Au cours de cette séance, nous allons préparer les jeux de données pour nos analyses et y appliquer quelques algorithmes de fouille de données.

1 Introduction

Nous allons construir des jeux de données à partir des séries historiques prédisposées lors de la séance précédente. Ensuite, nous essayerons d'étudier si ces séries présentent des patrons cachés qui rendraient possible des « arbitrages stochastiques ». De plus, nous irons tester si ces patrons deviennent plus faciles à trouver si l'on change de dévise de référence.

Nous nous servirons encore du système R pour manipuler les données. En outre, nous appliquerons quelques méthodes implantées dans Weka.

2 Ressources nécessaires

Nous allons utiliser les fichiers préparés lors de la première séance de TD.

3 Transformation

Pour nos analyses, il nous faudra des jeux de données constitués de lignes qui seront des points dans un espace des phases reconstruit, ayant la forme générale

$$x_1, x_2, \ldots, x_n, y$$

où y peut être la caractéristique à prédire (par exemple, prix à $t+1$) ou la classe à laquelle la ligne appartien (par exemple, *long*, *neutral*, *short*).

Nous construirons quelques exemples de ces jeux de donnees :

1. Des jeux de données pour la série $\{x_t\}_{t=0,\ldots,T}$ où la ligne pour la période t a la forme

$$\ln\frac{x_t}{x_{t-k}}, \ldots, \ln\frac{x_t}{x_{t-2}}, \ln\frac{x_t}{x_{t-1}}, \ln\frac{x_{t+1}}{x_t},$$

pour $k = 5, 20, 200$. Suggestion : écrivez une fonction `lagged.logret(x, k)`, qui, étant donnée le vecteur x et le décalage maximum k, renvoie un `data.frame` contenant le jeu de données demandé. La colonne des rendements logarithmiques décalés de i périodes peut être obtenu en un seul coup grâce à l'expression

```
log(x[(k + 2):length(x)]/x[(k - i + 1):(length(x) - i - 1)])
```

2. Un jeu de données pour la série $\{x_t\}_{t=0,\ldots,T}$ où la ligne pour la période t a la forme

$$\ln \frac{x_t}{\mathrm{SMA}(t,200)}, \ln \frac{x_t}{\mathrm{SMA}(t,50)}, \ln \frac{x_t}{\mathrm{SMA}(t,20)}, \ln \frac{x_t}{\mathrm{SMA}(t,5)}, \ln \frac{x_{t+1}}{x_t},$$

où $\mathrm{SMA}(t,k)$ est la moyenne glissante de x à k périodes, calculée à la période t.

La *moyenne glissante simple* d'une série temporelle $\{x_t\}$ est à son tour une série temporelle $\{m_t\}$, dont chaque élément réprésente la moyenne des k éléments précédents dans la série $\{x_t\}$:

$$m_t = \frac{1}{n} \sum_{i=0}^{k-1} x_{t-i}. \tag{1}$$

On dit alors que k est l'*ordre* de la moyenne glissante.

Suggestion : écrivez une fonction `SMA(x, t, k)` qui renvoie le vecteur des moyennes glissantes de la série x ; ensuite, écrivez une fonction `lagged.SMA(x)`, qui, étant donnée le vecteur x, renvoie un `data.frame` contenant le jeu de données demandé.

3. La même chose, mais cette fois-ci avec des moyennes glissantes exponentielles avec $\alpha = 1/10, 2/10, \ldots, 9/10$.

La *moyenne glissante exponentielle* d'une série temporelle $\{x_t\}$ est une série temporelle $\{m_t\}$ dont chaque élément m_{t_0} réprésente la moyenne pondérée de tous les éléments précédents dans la série $\{x_t\}$ pondérés pour un facteur qui diminue exponentiellement au fur et à mesure que t remonte vers le passé :

$$m_t = \alpha \sum_{n=0}^{\infty} (1-\alpha)^n x_{t-n}. \tag{2}$$

La moyenne glissante exponentielle peut aussi être exprimée en fonction de cette même moyenne calculée lors de la précédente période :

$$m_t = \alpha x_t + (1-\alpha) m_{t-1}, \tag{3}$$

ce qui rend son calcul beaucoup plus aisé.

Suggestion : écrivez une fonction `EMA(x, t, alpha)` qui renvoie le vecteur des moyennes glissantes exponentielles de la série x ; ensuite, écrivez une fonction `lagged.EMA(x)`, qui, étant donnée le vecteur x, renvoie un *dataframe* contenant le jeu de données demandé.

4. Pour les trois cas considérés ci-dessus, on remplacera les valeurs de la dernière colonne par l'une des trois catégories
   — *long*, si $\ln \frac{x_{t+1}}{x_t} > \frac{1}{250}$;
   — *neutral*, si $\left|\ln \frac{x_{t+1}}{x_t}\right| \leq \frac{1}{250}$;
   — *short*, si $\ln \frac{x_{t+1}}{x_t} < -\frac{1}{250}$.

   Suggestion : en R, vous pouvez vous servir d'une expression comme la suivante :

```
ifelse(abs(dataset$NextLogRet) <= threshold,
       "neutral",
       ifelse(dataset$NextLogRet < 0, "short", "long"))
```

   où `threshold` $= \frac{1}{250}$ et `dataset` est un *dataframe* contenant le jeu de données, avec la dernière colonne nommée `NextLogRet`.

   Pensez à sauvegarder les deux versions de chaque jeu de donnée (celle avec la dernière colonne numérique, adaptée pour faire de la prédiction, et celle avec la dernière colonne catégorielle, adaptée pour faire de la classification) dans des fichiers CSV distincts, l'un avec le suffixe `-num.csv` et l'autre `-cat.csv`. Pensez aussi à coder les noms des fichiers de façon systématique.

4 Classification

Nous allons utiliser quelques méthodes implantées dans Weka pour faire de la classification et/ou prédiction sur les jeux de données construits dans l'étape précédente. Installez (si vous ne l'avez pas encore fait) et démarrez Weka.

4.1 Arbres de décision

1. Ouvrez la fenêtre « Explorer » de Weka et chargez le fichier CSV que vous souhaitez utiliser pour la classification. Pensez à choisir un fichier avec la dernière colonne catégorielle.
2. Dans l'onglet « Classify », choisissez la méthode de construction d'arbres de décision « J48 » (ça se trouve sous « classifiers/Trees »).
3. Sans modifier les paramètres et les options proposés par défaut pas Weka, démarrez la classification en appuyant sur le bouton « Start ».
4. Il est possible d'utiliser les jeux de données avec la dernière colonne numérique aussi. Dans ce cas, il faut appliquer le filtre « Discretize » pour transformer la dernière colonne en une colonne catégorique par discrétisation (suggestion : il faut dire à ce filtre d'ignorer la classe—plus d'explications dans l'aide en ligne du filtre).
5. Analysez et comparez les résultats que vous obtenez avec les différents jeux de données.

4.2 Naïve Bayes

Faites la même chose avec la méthode « NaiveBayes » (sous « classifiers/Bayes »). Comparez les résultats à ceux que vous avez obtenus grâce aux arbres de décisions.