Séries Chronologiques

Université d'Angers – M2 Data Science

Fiche de TP

TP 1. *(Décomposition additive – R)* L'objectif de ce premier TP est la manipulation de la fonction `decompose` de R. On commencera par récupérer le jeu de données `AirPassengers` du package `datasets` puis, à l'aide d'une représentation graphique, par justifier la pertinence d'une transformation de données avant la modélisation additive. On prendra le temps de comprendre le fonctionnement de la méthode, en particulier les variables en sortie (`trend`, `seasonal`, `figure`, `random`). Afin de proposer un modèle prédictif, on pourra poser un modèle de régression linéaire sur la tendance de long terme puis conclure en prédisant deux nouvelles périodes (ne pas oublier que la série doit être ramenée à son échelle d'origine). Faire des représentations graphiques.

TP 2. *(Facebook Prophet – Python)* Dans cet exercice, on va travailler avec Python sous la forme d'un notebook (on évitera Spyder, car sa fenêtre graphique n'est pas toujours compatible avec les packages que l'on va utiliser). L'objectif sera de mettre en pratique le modèle *Facebook Prophet* (voir la remarque en page 6 du poly) à l'aide d'un tutoriel.

- Commencer par récupérer le dataset `Crypto.csv` puis le charger avec `pandas`, et observer la structure des données. Il s'agit du cours horaire du bitcoin BTC/USD sur la période 07/2017–10/2020.
- Convertir la colonne `Unix Timestamp` en `datetime` et remplacer l'ancienne colonne `Date` du dataset par celle-ci.
- Effectuer la représentation en *candlestick* (module `plotly.graph_objs`) spécifique aux indices boursiers, avec les colonnes `Open`, `High`, `Low` et `Close`. Zoomer et observer.
- Ajouter une nouvelle colonne `TypPrice` au dataset qui contiendra le *prix typique* de l'action, moyenne des prix haut, bas et de clôture. Visualiser sur un même graphique l'évolution de `Low`, `TypPrice` et `High`. On pourra ajouter un *rangeslider* (module `plotly.offline`).
- On va maintenant placer un modèle de type *Facebook Prophet* sur le prix typique, c'est-à-dire un modèle additif de la forme
$$X_t = T_t + S_t + H_t + \varepsilon_t$$
où (T_t) est la tendance de long terme, (S_t) la saisonnalité multi-échelle (journalière, hebdomadaire et annuelle) et (H_t) un éventuel effet week-ends/jours fériés/vacances (que l'on n'exploitera pas ici). Récupérer le script correspondant et le faire tourner pour bien saisir ses capacités : estimations, prédictions, graphiques, validation croisée, etc.

TP 3. *(Simulations ARMA – R)* On souhaite se familiariser avec la manipulation des processus ARMA. L'objectif est également la maîtrise des outils principaux d'aide à la modélisation ARMA (corrélations empiriques, tests de stationnarité, tests de bruit blanc, etc.). Quelques suggestions :

- Pour bien saisir le principe de l'ARMA, programmer une fonction
```
simulerARMA(n, m, Phi, Theta, s2)
```
qui simule une série chronologique de taille `n` selon un modèle ARMA centré en `m`, dont les coefficients AR sont dans le vecteur `Phi` et les coefficients MA dans le vecteur `Theta`, et dont le bruit est gaussien de variance `s2`. Cette fonction représentera la série graphiquement et la renverra sous forme de vecteur numérique. (Suggestion annexe : comparer votre fonction avec `arima.sim` du package `tseries`).

— Observer la différence graphique entre les ARMA stationnaires et les ARMA non stationnaires, par exemple avec `Phi = c(1)` et `Phi = c(-1)`.
— Étudier les ACF et PACF en sortie des simulations ARMA (commandes `acf` et `pacf`). En particulier, observer le comportement des ACF sur les MA et des PACF sur les AR (stationnaires) en rapport avec les Prop. 2.6 et 2.9 du cours.
— Lire la documentation associée aux tests de Ljung-Box (`Box.test` du package `stats`), et aux tests de stationnarité (`adf.test` et `kpss.test` du package `tseries`). Effectuer quelques tests sur vos simulations.
— Lire l'exemple en haut de la p. 37 du cours : comprendre le fonctionnement de la commande `Arima` du package `forecast` et l'utilité de ses arguments principaux, et tenter de reproduire l'expérience (il existe aussi une fonction `arima` dans le package `stats` qui travaille de la même manière). On pourra en profiter pour étudier la commande bien pratique `auto.arima` de ce même package.
— Charger et observer le jeu de données `electricity` du package `TSA`, puis considérer la série transformée $Y_t = \ln(X_t)$. Récupérer les fluctuations issues du `decompose` puis chercher un ARMA adapté à sa modélisation. Représenter graphiquement la superposition du signal et de sa modélisation (les valeurs reconstruites par le modèle sont accessibles par la commande `fitted`).

TP 4. *(Diagnostic – R)* Une fois la modélisation chronologique effectuée, il est en général souhaitable de s'assurer que les résidus forment un bruit blanc (afin de vérifier qu'on n'a pas oublié d'extraire des corrélations dans la série) et qu'ils sont approximativement gaussiens (car l'estimation ARMA se fait sous l'hypothèse de normalité de la série). Programmer une fonction

`checkupRes(Res)`

qui prend en charge une série de résidus `Res` et affiche des graphiques selon la disposition

Graph. 1		
Graph. 2	Graph. 3	Graph. 4
Graph. 5	Graph. 6	Graph. 7

qui peut être obtenue par la commande `layout(matrix(c(1,1,1,2:7), nrow=3, ncol=3, byrow=TRUE))`. Ces graphiques sont respectivement :
— Graph. 1 : évolution de la série.
— Graph. 2 : ACF.
— Graph. 3 : PACF.
— Graph. 4 : nuage de points `Res[i]` en fonction de `Res[i-1]`.
— Graph. 5 : histogramme.
— Graph. 6 : QQ plot par rapport aux quantiles gaussiens.
— Graph. 7 : nuage de points de la série renormalisée (centrée/réduite), muni des horizontales ± 1.96.

Ces critères étant purement visuels, il est évident que l'on attend un affichage soigné et lisible des graphiques. L'objectif est également de pouvoir réutiliser cette fonction dans les futurs diagnostics (TP et projet).

TP 5. *(Modèles ARMA – R)* Au choix : télécharger sur le site `https://finance.yahoo.com/` le cours du Bitcoin USD (BTC-USD) avec un pas journalier (option 'daily' dans l'onglet 'Historical Data') ou bien récupérer le dataset `BTC-USD.csv`.
— Prendre le logarithme de la série, puis observer graphiquement l'évolution depuis le début de décembre 2021. Cela paraît-il stationnaire ? Confirmer cette intuition par les tests appropriés.
— Supprimer la tendance linéaire décroissante, soit 'manuellement' (par une régression linéaire), soit par l'utilisation de `include.drift` dans la commande `Arima`.
— Proposer un modèle ARMA. On pourra se fier à plusieurs indicateurs : comportement des ACF/PACF, AIC/BIC, significativité des estimateurs, `auto.arima`, étude des résidus, etc.
— Superposer la série initiale avec les valeurs reconstruites par le modèle munies d'un intervalle de confiance.

TP 6. *(Modèles ARMA – R)* On s'intéresse dans cet exercice à deux séries de données réelles à modéliser par un ARMA.

- Récupérer la série `precip1.dat` mesurant les précipitations annuelles (en inches) à Londres entre 1813 et 1912 : `Rain = scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat", skip=1)` à convertir au format `ts` : `Rain = ts(Rain, start=c(1813))`.
- Appliquer la méthodologie du raisonnement précédent. Quel modèle proposer ?
- Même question avec la série `dvi.dat` contenant des données volcaniques de l'hémisphère nord de 1500 à 1969 : `Volcano = scan("http://robjhyndman.com/tsdldata/annual/dvi.dat", skip=1)` à convertir au format `ts`. Il faudra ici réfléchir à une transformation de données adéquates.
- Qu'observe-t-on au niveau des pics ?

TP 7. *(Modèles ARIMA – R)* On va tenter de modéliser le jeu de données `gold` du package `forecast`. Après l'avoir récupéré et avoir lu sa description (dans le `help`), on le tronquera à sa dernière année (`X = gold[(n-364):n]`) pour ne conserver que la partie post-rupture que vous pouvez observer sur le signal complet. Ces observations portent donc sur la période du 01/04/88 au 31/03/89. Quelques suggestions :

- Tout d'abord, comme des valeurs sont manquantes, on pourra les reconstruire par une simple interpolation linéaire. Cette méthode un peu simpliste est à éviter pour les séries saisonnières où l'information est à aller chercher dans le motif périodique, mais ici cela semble raisonnable.
- Vérifier graphiquement que la série présente une tendance linéaire décroissante et qu'elle ne peut pas être considérée comme stationnaire (les tests ADF et KPSS devraient vous le confirmer).
- Pour mettre en place un modèle ARIMA, on commence par différencier la série avec `DX = diff(X)` pour s'assurer ensuite que les incréments obtenus sont stationnaires. On cherche alors un modèle de la forme $(X_t) \sim \text{ARIMA}(p, 1, q)$. Justifier graphiquement la pertinence du choix $p = 0$ et $q = 1$ à travers les ACF/PACF des incréments.
- Poser un modèle ARIMA(0,1,1) sur la série initiale, avec tendance linéaire (`include.drift=TRUE`). Vérifier que les résidus sont raisonnables (bruit blanc avec normalité douteuse... en raison de quelques outliers). Le bon moment pour utiliser votre `checkupRes` ?
- Effectuer une prédiction pour le mois d'avril 1989. Superposer le signal de départ ainsi que votre prédiction munie d'un intervalle de prédiction à 80% (voir la documentation de la commande `forecast`).

TP 8. *(Modèles ARMAX – Python)* Dans une modélisation ARMAX (X pour *eXogenous*), on insère dans un ARMA une contribution exogène sous la forme d'une régression linéaire. Par exemple, un ARMAX(2, 1) avec $(Z_{1,t})$ et $(Z_{2,t})$ deux processus évoluant parallèlement à (X_t) pourrait s'écrire

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \beta_1 Z_{1,t} + \beta_2 Z_{2,t} + \theta_1 \varepsilon_{t-1} + \varepsilon_t.$$

Il est même possible d'envisager des autorégressions dans le processus exogène en posant $Z_{2,t} = Z_{1,t-1}$, etc. En pratique, l'estimation se fera par une régression linéaire avec erreurs ARMA (ce qui formellement ne correspond pas tout à fait au modèle écrit ci-dessus). La première question est donc de savoir jusqu'à quel point (X_t) et les processus exogènes sont corrélés, ce qui fait intervenir la notion de *corrélation croisée*.

- Récupérer avec `pandas` le dataset `ConsoTemp.csv` contenant la consommation électrique journalière à l'échelle d'une ville en fonction de la température.
- Étudier les corrélations croisées entre les deux séries avec `ccf` de `statsmodels.tsa.stattools`.
- Modéliser la consommation électrique par un ARMAX en exploitant l'argument `exog` de la méthode `SARIMAX` de `statsmodels.tsa.statespace.sarimax`.
- Effectuer quelques représentations graphiques appropriées.

TP 9. *(Modèles SARIMA – Python)* On va traiter le jeu de données lié aux émissions de CO_2 disponible dans `statsmodels.api.datasets`. Pour vous aider à bien saisir les opérations demandées, un script Python est fourni. On proposera de plus des représentations graphiques adaptées et les tests usuels de diagnostic (stationnarité, blancheur, normalité).

- Pour simplifier, tronquer la série sur la période 1990-1999, et vérifier que l'on peut raisonnablement retenir une périodicité de 52 sur ce jeu de données (mesures hebdomadaires sur cycle annuel).

— Mettre en place une décomposition additive.
— Mettre en évidence le fait qu'il faudra probablement retenir $d = 1$ et $D = 1$ dans une modélisation SARIMA.
— Tester un SARIMA$(2, 1, 2) \times (1, 1, 1)_{52}$ avec tendance linéaire puis, après avoir constaté un manque de significativité de certains estimateurs, un SARIMA$(1, 1, 1) \times (1, 1, 1)_{52}$.
— Utiliser ce modèle pour reprédire l'année 2000.

TP 10. *(Modèles SARIMA – R)* Cet exercice est également dédié aux modèles SARIMA, fondamentaux en analyse chronologique. On se relance dans la modélisation de la série `AirPassengers` du package `datasets`, l'objectif étant de reprogrammer et de comprendre les sorties présentées en cours (Sec. 4.2).
— Créer le vecteur `LAP`, logarithme de la série initiale, puis étudier la présence de corrélation, de périodicité et de stationnarité avant et après les différenciations $I - B$, $I - B^{12}$ et $(I - B)(I - B^{12})$.
— Modéliser cette série par un SARIMA$(1, 0, 1) \times (0, 1, 1)_{12}$ avec tendance linéaire. On utilisera pour cela la commande `Arima` du package `forecast`.
— Chercher avec la commande `auto.arima` le SARIMA sans tendance le mieux adapté sur la base du BIC. On retiendra $p, q \leq 5$ et $P, Q \leq 1$.
— Comparer ces deux modèles sur la base du critère prédictif suivant : on retire la dernière période $(X_{n-s+1}, \ldots, X_n)$, on la reprédit par $(\widehat{X}_{n-s+1}, \ldots, \widehat{X}_n)$ et on calcule l'écart quadratique moyen entre observations et prédictions,

$$\text{MSE} = \frac{1}{s} \sum_{k=1}^{s} (X_{n-k+1} - \widehat{X}_{n-k+1})^2.$$

On prendra garde à bien estimer les modèles sur la série tronquée (pour éviter le biais dû à l'utilisation d'une donnée connue pour la reprédire).
— Représenter graphiquement la série initiale munie des prédictions que vous retenez pour les deux périodes suivantes ($h = 24$), ainsi qu'un intervalle de prédiction à 95%. On pensera à corriger la prédiction lors du passage à l'exponentielle.

TP 11. *(Rupture, SARIMA et Lissages – Python)* On présente dans cet exercice une méthode très empirique (quoique fonctionnelle) pour détecter une rupture dans une évolution chronologique. La suite et fin de l'exercice consiste à faire de la prédiction à l'aide d'un SARIMA et d'un lissage de Holt-Winters, une méthode qui sera rapidement introduite.
— On va travailler sur la série `SNCF.csv` fournie sur Moodle. Charger la série à l'aide de `pandas`, la représenter graphiquement et identifier la présence d'une rupture.
— Après avoir passé la série au log, on va mettre en place deux modèles de régression : une régression constante sur $(X_1, \ldots, X_R)$ et une régression linéaire sur $(X_{R+1}, \ldots, X_n)$. Localiser l'abscisse R de rupture en minimisant le MSE issu de ces régressions (`LinearRegression` de `sklearn.linear_model` et `mean_squared_error` de `sklearn.metrics`).
— Soit maintenant la série tronquée $(X_{R+1}, \ldots, X_n)$ de taille $n - R$, qui devient la nouvelle valeur de n. Identifier deux modèles SARIMA et les comparer sur la prédiction de la dernière période connue. On pourra par exemple utiliser `auto_arima` de `pmdarima.arima` avec $(d, D) = (0, 1)$ et $(d, D) = (1, 1)$.
— Les lissages exponentiels sont des alternatives déterministes pour effectuer des prédictions à court terme sur une série chronologique, rapides en temps de calcul. En contrepartie, ils sont souvent un peu simplistes mais arrivent à donner des résultats intéressants. Le lissage de Holt-Winters périodique et additif est défini par

$$\forall\, \tau \,\leq\, t \,\leq n, \quad \left\{ \begin{array}{rcl} c_t & = & \alpha\,(X_t - s_{t-\tau}) + (1-\alpha)\,(c_{t-1} + \ell_{t-1}) \\ \ell_t & = & \beta\,(c_t - c_{t-1}) + (1-\beta)\,\ell_{t-1} \\ s_t & = & \gamma\,(X_t - c_t) + (1-\gamma)\,s_{t-\tau} \end{array} \right.$$

où α, β et γ sont des paramètres à déterminer, τ est la période, et où les valeurs initiales sont à fixer. La prédiction à l'horizon h vaut alors

$$\widehat{X}_{n+h} = c_n + h\,\ell_n + s_{n-\tau+h}.$$

Comparer les modèles SARIMA retenus avec un lissage de Holt-Winters (`ExponentialSmoothing` de `statsmodels.tsa.holtwinters`).

— Prédire deux années supplémentaires de la série.

TP 12. *(Modèles GARCH – R)* On va proposer dans ce dernier exercice un survol très rapide de la modélisation GARCH. Les étudiant-e-s intéressé-e-s, particulièrement en cas de stage dans le domaine financier, sont encouragé-e-s à s'auto-former à la pratique plus poussée de ces modèles conditionnellement hétéroscédastiques.

— Charger le jeu de données `EuStockMarkets` du package `dataset`, contenant l'évolution de 4 indices boursiers européens, et conserver la première série numérique (DAX), appelée (p_t) dans la suite.

— On veut étudier l'évolution du rendement $r_t = \frac{p_t - p_{t-1}}{p_{t-1}}$. Justifier l'approximation

$$X_t = \ln \frac{p_t}{p_{t-1}} \approx r_t$$

et donc la pertinence de travailler sur les incréments de la série logarithmique. Constater empiriquement le comportement bruit blanc de (X_t) et le comportement ARMA de (X_t^2).

— Pour mettre en évidence la présence d'hétéroscédasticité conditionnelle, représenter la suite des variances empiriques du signal avec une fenêtre glissante de `fen` valeurs (c'est-à-dire, en considérant des tronçons successifs de taille `fen`, à fixer par exemple à 100).

— Recentrer la série puis mettre en place un modèle GARCH$(1,1)$ et un modèle GARCH$(0,2)$ à l'aide de la commande `garch` du package `tseries`. Comparer ces deux modèles sur la base du BIC.

— Superposer sur un même graphique l'évolution de la série, l'estimation de la volatilité et celle de l'écart-type en régime stationnaire (± 1 écart-type), par le modèle de votre choix. Comparer avec l'intuition visuelle.