M1 ISMAG
MIS243Y - Séries chronologiques
TP 1 - Premières manipulations des processus AR/MA/ARMA

---

1 Simulation de processus

1.1 Simulation d'un processus MA

Pour simuler simplement T réalisations d'un processus moyenne mobile d'ordre q ayant les paramètres θ_1, ..., θ_q et dont la variance des innovations est σ^2, on peut utiliser l'algorithme suivant :

1. Générer des innovations i.i.d. $(\eta_t)_{t=1,\dots,T}$.
2. Calculer récursivement les valeurs

$$X_t = \eta_t - \sum_{k=1}^{q} \theta_k \eta_{t-k}. \tag{1}$$

Par exemple, supposons que l'on veuille générer $T = 1000$ réalisations du processus MA(1) suivant

$$X_t = \eta_t - \frac{1}{3}\eta_{t-1}, \quad (\eta_t) i.i.d. \sim \mathcal{N}(0,1).$$

Il faut tout d'abord introduire quatre éléments :
- `theta`, un vecteur de longueur 1 comportant les paramètres θ_1 ;
- `innov`, le vecteur des innovations de longueur $T+1$;

puis définir X_t selon la récurrence (1).

Écrire les lignes de commande correspondant à cet algorithme.

1.2 Simulation d'un processus AR

Pour simuler simplement T réalisations d'un processus autorégressif d'ordre p ayant les paramètres ϕ_1, ..., ϕ_p et dont la variance des innovations est σ^2, on peut utiliser l'algorithme suivant :

1. Fixer p valeurs initiales réelles arbitraires X_0, ..., X_{p-1}.
2. Générer des innovations i.i.d. $(\eta_t)_{t=1,\dots,T+T_0}$.
3. Calculer récursivement les valeurs

$$X_t = \sum_{k=1}^{p} \phi_k X_{t-k} + \eta_t. \tag{2}$$

4. Éliminer les T_0 premières valeurs ainsi générées.

En pratique, les ordres de grandeur sont de $T = 1000$ et $T_0 = 200, 300$ ou 500 (selon la distance, dans le plan complexe, entre les racines du polynôme associé aux coefficients et le cerle unité).

Par exemple, supposons que l'on veuille générer $T = 1000$ réalisations du processus AR(2) suivant

$$X_t - \frac{4}{15}X_{t-1} + \frac{1}{15}X_{t-2} = \eta_t, \quad (\eta_t) i.i.d. \sim \mathcal{N}(0,1).$$

Il faut tout d'abord introduire quatre éléments :

– `phi`, un vecteur de longueur p comportant les paramètres $\phi_1, \ldots, \phi_p$;
– `init`, un vecteur contenant p valeurs arbitraires pour commencer l'algorithme ;
– `innov`, le vecteur des innovations de longueur $T + T_0$;
– `t`, qui est le nombre T_0 de valeurs à rejeter à la fin de la procédure.

puis définir X_t selon la récurrence (2).

On écrira donc les lignes de commande suivantes :

```
T=1000
T0=500
phi=c(4/15,-1/15)
p = length(phi)
init=rep(1,2)
innov=rnorm(T+T0)
n = length(innov)
x = init

for (i in (p+1) :n)
{
aux = innov[i] + sum(phi*x[(i-1) :(i-p)])
x = c(x,aux)
}
x = x[(T0+1) :n]
```

1.3 Simulation d'un processus ARMA

Plus généralement, nous allons nous intéresser aux processus ARMA : ce sont des processus ayant une partie AR et une partie MA i.e. de la forme

$$X_t - \phi_1 X_{t-1} - \phi_2 X_{t-2} \ldots \phi_p X_{t-p} = \eta_t - \theta_1 X_{t-1} - \theta_2 X_{t-2} \ldots \theta_q X_{t-q}, \quad (\eta_t) i.i.d. \sim \mathcal{N}(0,1).$$

R possède une fonction pour simuler directement des processus ARMA (ou ARIMA) : `arima.sim`. Nous allons à présent étudier les paramètres principaux pour utiliser cette fonction. Pour simuler un processus ARMA, on a besoin d'au moins trois éléments : un modèle, un nombre de réalisations et un processus d'innovations.

Par exemple, supposons que l'on veuille simuler le processus ARMA$(2,1)$ stationnaire et causal

$$X_t - X_{t-1} + \frac{1}{4} X_{t-2} = \eta_t - \eta_{t-1}, \quad (\eta_t) i.i.d. \sim \mathcal{N}(0,1).$$

La spécification d'un tel modèle se fait à partir d'une liste composée de deux vecteurs :
– le vecteur des coefficients autorégressifs ;
– le vecteur des coefficients du processus à moyenne mobile.

La simulation du processus en R se base sur la syntaxe suivante :

```
arima.sim(model = monmodele,n,rand.gen, innov = rand.gen(n, ...), n.start,...)
```

avec

`model`	liste des paramètres de la partie AR et de la partie MA. Une liste vide correspond au modèle ARIMA(0,0,0) qui est un bruit blanc.
`n`	nombre de réalisations désirées ($n = 100$ par défaut)
`rand.gen`	paramètre optionnel qui permet de spécifier quel est le processus des innovations
`innov`	paramètre optionnel des innovations
`n.start`	nombre d'onservations initiales à supprimer
...	

Remarque : *On retrouve en fait dans cette fonction la syntaxe et les paramètres utilisés pour la fonction construite au paragraphe précédent pour générer des processus autorégressifs. Il y a encore d'autres paramètres facultatifs à insérer dans l'argument de la fonction* ***`arima.sim`****. Nous ne les étudierons pas ici en détail, mais on peut les consulter en lisant la rubrique d'aide sur cette fonction.*

Pour l'exemple, on tapera

```
monmodele = list(ar=c(1,-.25),ma=1)
monprocessus=arima.sim(model=monmodele,n=100,rand.gen=rnorm)   pour le générer
plot.ts(monprocessus)                                          pour le représenter
```

Exercice 1 : A l'aide de la fonction R `arima.sim`, simuler

1. les processus MA(1) et AR(2) des paragraphes précédents
2. un processus AR(2)
3. un processus MA(3)
4. un processus ARMA(2,2)

On choisira dans chaque cas les paramètres.

2 Représentations graphiques

Avant d'appliquer les méthodes d'estimation et de sélection de modèle dans le but de faire de la prédiction, il convient de représenter la série chronologique observée et de faire une première analyse de ses éventuelles tendances, saisonnalités et autres particularités. Représentons la série simulée précédemment :

```
plot.ts(X)
```

Exercice 2 : Représenter les séries de la section précédente.

La tendance ou la saisonnalité d'une série sont en général détectées à partir du graphe de la série. Elles influencent cependant le comportement de la fonction d'autocorrélation : une tendance entraîne une persistance dans la fonction d'autocorrélation, et une autocorrélation pratiquement périodique est un indice pour une composante saisonnière. La détection d'une tendance ou d'une saisonnalité peut donc parfois être justifiée par le comportement de son autocorrélation.

Pour calculer et représenter la fonction d'autocorrélation empirique d'une série observée, il faut utiliser en R la fonction `acf()` où la série chronologique est placée en argument. Par défaut, le logiciel fournit la fonction d'autocorrélation pour une longueur proportionnelle au logarithme de

la série et représente graphiquement la fonction. On peut changer ces paramètres en utilisant la syntaxe suivante :

```
acf(serie,lag.max=n,plot=F)
```

où `serie` est la série observée, `n` est le nombre de valeurs désirées pour la fonction d'autocorrélation, et `plot=F` signifie que l'on ne désire pas de représentation graphique (par défaut, `plot=T`). Par défaut, la représentation graphique montre également l'intervalle de confiance au niveau 95%

$$\left[-1.96/\sqrt{T}; 1.96/\sqrt{T}\right]$$

pour un bruit blanc. Lorsque T est assez grand, les autocorrélations d'un bruit blanc sont approximativement indépendantes et de loi $\mathcal{N}(0, \frac{1}{T})$. Ainsi 95% des autocorrélaations devraient se trouver dans l'intervalle précédent.

En précisant à la fonction `acf()` un argument `type`, on peut obtenir les autocorrélations partielles :

```
acf(serie, type = "partial")
```

tandis qu'en écrivant `type="covariance"` à la place de cet argument, on obtient la fonction d'autocovariance :

```
acf(serie, type = "partial")
```

On peut aussi utiliser la commande `pacf` pour obtenir le corrélogramme partiel.

Remarque : Les paramètres graphiques et le niveau de l'intervalle de confiance peuvent être modifiés en utilisant la fonction `plot()` qui doit contenir une fonction d'autocorrélation en argument [1] : `plot(acf(serie))`

Cette fonction peut contenir tous les paramètres graphiques en argument ainsi que les deux arguments suivants :

`conf.int=T` ou `F`

selon que l'on veuille ou non que l'intervalle de confiance soit représenté (la valeur par défaut est `True`), et

`ci = 0.9`

où `0.9` ou toute autre valeur précise le niveau de l'intervalle de confiance (cet argument suppose bien entendu que `conf.int=T`).

Exercice 3 : Vérifier qu'une tendance entraîne une persistance dans la fonction d'autocorrélation alors que la périodicité entraîne plutôt une forme sinusoïdale.

Exercice 4 : A titre d'exemple, nous allons étudier le processus AR(2) suivant :

$$X_t - 0.9X_{t-1} + 0.8X_{t-2} = \eta_t.$$

1. Dans cette commande, on spécifie que `plot = F` car, sinon, R représente deux fois la fonction d'autocorrélation : la première fois à cause de l'appel `acf(serie)`, et la seconde par la commande `acf.plot()`

Commençons par le simuler en appliquant la fonction `arima.sim` :

```
X=arima.sim(100,model=list(ar=c(0.9,-0.8)))
```

Ensuite, on tape les lignes de commande suivantes :

`plot.ts(X)`	Représentation graphique de la série
`mean(X)`	Calcul de la moyenne empirique
`acf(X)`	Représentation du corrélogramme
`acf(X,type="covariance")`	Représentation du variogramme
`pacf(X)`	Représentation du corrélogramme partiel

1. Le processus est-il centré ?
2. Que peut-on dire de la tendance et de la saisonnalité de la série ?
3. Quel est le lien entre le corrélogramme et le variogramme ?
4. Commenter les corrélogrammes de la série $(X_t)_t$.

Pour expliquer les différents graphiques, on calcule le discriminant du polynôme Φ qui est négatif et vaut -2.39. Les racines sont donc complexes. Ceci explique la forme "sinusoïdale" du corrélogramme.
D'autre part, les corrélations d'ordre 1,3 et 6 apparaissent assez élevées en module. Cette propriété devrait réapparaître sur les trajectoires : une forte valeur à une certaine date devrait en général impliquer une faible valeur trois dates après et une forte valeur six dates après (le caractère faible ou fort dépend bien évidemment du signe de ρ).

5. Retrouve-t-on ce phénomène sur le graphe de la trajectoire ?

La présence du bruit de variance σ^2 explique évidemment que la fonction ne soit pas strictement périodique.

Exercice 5 : Simuler le processus ARMA(1,1) suivant :

$$X_t - 0.5X_{t-1} = \eta_t - 0.6\eta_{t-2}$$

Répondre aux questions de l'exercice précédent.

3 Choix de l'ordre et estimation des coefficients ARMA

Étudions dans cette section le processus ARMA$(2,1)$ défini par

$$X_t - X_{t-1} + \frac{1}{4}X_{t-2} = \eta_t - \eta_{t-1}, \quad (\eta_t) i.i.d. \sim \mathcal{N}(0,1).$$

On tapera donc les commandes :

```
monmodele = list(ar=c(1,-.25),ma=1)
X=arima.sim(model=monmodele,n=100,rand.gen=rnorm)   pour le générer
plot.ts(X)                                          pour le représenter
```

On va faire comme si on ne connaissait pas les coefficients et la variance mais seulement l'ordre

et on va procéder à l'estimation des paramètres. On peut ajuster un modèle ARIMA(p, d, q) à l'aide de la commande `arima`. Pour cela, on peut spécifier les ordres (voir le cours pour la détermination et l'estimation de ces ordres) ou laisser R libre de décider.

Dans le cadre de l'exemple, comme on souhaite ajuster un modèle ARMA(2,1) (ce qui revient à une ARIMA(2,0,1)), il suffit de lancer les commandes :

`X.ord=c(2,0,1)`	Spécifier le modèle
`X.arima=arima(X,order=X.ord)`	Estimer les paramètres du modèle

Recommencer avec une simulation plus grande ($n = 1000$ par exemple) et vérifier que les paramètres sont mieux estimés.

Remarque : Pour ajuster un modèle possèdant une composante à moyenne mobile, les estimateurs de Yule-Walker ne conviennent plus et il faut utiliser les estimateurs de moindre carrés ou de maximum de vraisemblance. Ces derniers estimateurs sont calculés en R par la même fonction `arima`. Pour sélectionner un modèle ARMA suivant la technique vue en cours, dans le cadre de la procédure de Box-Jenkins, il faut tout d'abord fixer des valeurs $p_{\max}$ et $q_{\max}$ maximales pour l'ordre du modèle à ajuster. Ces valeurs peuvent par exemple être choisies par l'observation des fonctions empiriques d'autocorrélation et d'autocorrélation partielle. A chaque valeur $1 \leq p \leq p_{\max}$, $1 \leq q \leq q_{\max}$, on peut utiliser la fonction `arima` pour calculer les estimateurs de maximum de vraisemblance des coefficients et la fonction AIC du modèle correspondant.

Voici une brève description des options de la commande `arima` :

x	la série à ajuster
order	pour indiquer l'ordre du processus ARIMA
seasonal	pour spécifier la partie saisonnière du modèle
include.mean	pour indiquer la présence d'une constante (T) ou pas (F) dans le modèle
method	méthode d'estimation : CSS-ML, ML, CSS

La méthode ML est celle du maximum de vraisemblance et la méthode CSS est celle qui minimise la sommes des carrés résiduels. Par défaut, la méthode CSS-ML est employée et conjugue ces deux méthodes : d'abord, elle minimise la somme des carrés résiduels, puis elle applique la méthode du maximum de vraisemblance. L'objet renvoyé par cette fonction possède, entre autres, les composantes suivantes :

`coeff`	estimateurs des coeficients des parties AR et MA, ainsi que de la constante
`sigma2`	estimation de la variance du bruit d'innovation
`var.coef`	matrice asymptotique de variance covariance pour les estimateurs de maximum de vraisemble
`loglik`	log-vraisemblance à l'optimum
`aic`	valeur du critère AIC pour le modèle `model` (si la méthode du maximum de vraisemblance est appliquée)
`resiudals`	estimation des innovations

4 Cas particulier : choix de l'ordre et estimation des coefficients AR

On peut ajuster de manière automatique un modèle autorégressif via la commande `ar`

```
serie.ar= ar(dserie)
```

qui a pour argument la série observée et renvoie un objet de type `liste` contenant :

`order`	ordre $\hat{p}$ du modèle ajusté et correspond à la minimisation de l'AIC
`ar`	estimation des coefficients du modèle autorégressif
`x.mean`	estimation de la moyenne de la série
`var.pred`	estimation de la variance des innovations
`aic`	`aic=T` signifie que l'on choisit le meilleur modèle selon le critère AIC tandis que `aic=F` signifie que l'on n'utilise pas le critère AIC. Il faut donc dans ce cas-là donner un ordre pour le modèle spécifié par la commande `ordre.max`.
`n.used`	nombre d'observations
`order.max`	ordre maximal considéré par la fonction AIC. Sa valeur par défaut est $\min(n.used - 1, 10\log_{10}(n.used))$, mais peut être spécifiée en introduisant un argument supplémentaire dans l'appel de la fonction, suivant la syntaxe : `ar(serie, order.max=N)` où N indique la longueur du vecteur `aic` à considérer
`partialacf`	vecteur de longueur `order.max` possédant les valeurs de la suite de la fonction d'autocorrélation partielle empirique
`resid`	objet de type `time series` contenant les résidus et du modèle ajusté aux données
`method`	chaîne de caractères (`string`) `yule-walker`
`series`	nom de la série introduite en argument de la fonction

On peut également choisir d'utiliser la fonction d'ajustement `ar` en imposant l'ordre du modèle à ajuster, sans passer par le critère de minimisation de l'AIC. En demandant

```
ar(dserie, aic=F, order.max=N)
```

R ajustera un modèle AR(N) à `dserie`.

Les coefficients sont estimés à l'aide des équations de Yule-Walker. Le modèle retenu est celui qui maximise le critère AIC (sauf si on ajoute l'option `aic=F` auquel cas la fonction ajuste un modèle autorégressif à l'ordre `ordre.max`). Cette fonction est en fait la même que la fonction `ar.yw`. Il existe trois autres fonctions qui utilisent une autre méthode pour estimer les paramètres du modèle : la fonction `ar.mle` qui repose sur les estimateurs du maximum de vraisemblance, `ar.burg` qui utilise l'algorithme de Durbin-Levinson et `ar.ols` qui résoud un problème de moindres carrés.

5 Choix de modèle et validation

Comme nous le verrons en cours, dans le cadre de la procédure de Box-Jenkins, il s'agit de comparer les différents ARMA envisagés (modèles différents (AR,MA, ARMA) et ordres différents) puis de sélectionner un modèle ARMA parmi plusieurs en se basant sur le critère AIC donnée en sortie de la fonction. Pour cela, on appliquera donc plusieurs fois la commande `arima` avec

des ordres différents.

Avant de conclure sur le choix d'un modèle, il est nécessaire d'effectuer des tests sur les résidus. Pour cela, on peut utiliser la commande `tsdiag` qui renvoie un graphique contenant de haut en bas : les résidus normalisés, la fonction d'autocorrélation des résidus et les p-valeurs de la statistique de Ljung-Box en fonction des décalages. Le test de Box-Ljung ainsi que celui de Box-Pierce (qui testent la non-corrélation) peuvent être aussi effectués en utilisant la commande Box.test qui prend en entrée la série de données initiales :

```
tsdiag(X.arima)
Box.test(X.arima$residuals,lag=1,type=c("Box-Pierce"))
Box.test(X.arima$residuals,lag=1,type=c("Ljung-Box"))
```

Avec la fonction `Box.test`, il faut préciser l'ordre du décalage et le type de test choisi. L'objet renvoyé par cette fonction possède, entre autres, les composantes suivantes :

`statistic`	valeur de la statistique de test
`parameter`	nombre de degrés de liberté de la loi du χ^2 (loi asymptotique)
`p.value`	p-valeur associée à la statistique de test.

- Commenter les résultats obtenus à l'aide de la fonction `tsdiag`.

6 Prédiction des processus ARMA

Une fois un modèle retenu, il est possible de faire des prévisions à horizons donnés en appliquant la fonction `predict`. Une fois un modèle retenu, il est possible d'appliquer la fonction `predict` pour effectuer une prévision à horizon donné. On peut aussi appliquer la théorie de la prédiction linéaire pour trouver un intervalle de confiance pour les valeurs futures de la série. La fonction à utiliser à cet effet est

```
predict(model, n)
```

où les arguments placés entre parenthèses sont obligatoires.

`serie`	série stationnaire observée
`model`	liste contenant les coefficients estimés
`n`	nombre de prédictions à estimer

Le résultat de cette fonction est une liste comportant deux éléments :

`mean`	valeur moyenne estimée de chaque prévision
`std.err.`	estimateur de $\sigma_\eta(.)$ qui intervient dans le calcul de l'intervalle de prévision $\left[\hat{X}_T(h) \pm z_{1-\alpha/2}\hat{\sigma_\eta}(h)\right]$, $1 \leq h \leq n$ qui, rappelons-le, ne tient pas compte de l'erreur d'estimation des paramètres du modèle ajusté

Pour avoir une prédiction à l'horizon 10, il suffit de taper la commande :

```
X.pred=predict(X.arima,10)
```

tandis que pour avoir les 10 prochaines prédictions, il faudra taper

```
X.pred=predict(X.arima,n.ahead=10)
```

Exercice 6
Soit le processus

$$X_t - X_{t-1} + \frac{1}{2}X_{t-2} - \frac{1}{3}X_{t-3} = \eta_t$$

avec η_t un bruit blanc gaussien centré réduit.

1. Après avoir identifié ce processus, simuler 50 réalisations de longueur 105.
2. Pour chaque simulation, extraire les 100 premières valeurs et estimer les paramètres d'un AR(3).
3. Pour chaque simulation, prédire les cinq valeurs suivantes.
4. Donner une estimation de l'erreur moyenne (biais) et de la variance de l'erreur de prévision à 1, 2, 3, 4 et 5 pas.
5. Recommencer en rallongeant la durée d'observation, et comparer aux résultats précédents.

M1 ISMAG
MIS243Y - Séries chronologiques
TP 2 - Manipulation complète d'une série chronologique

Nous avons vu dans le dernier TP comment simuler des processus ARMA puis les différentes étapes du traitement d'une série temporelle :

- la représentation graphique ;
- la détermination des fonctions d'autocorrélation et d'autocorrélation empiriques ;
- le choix de l'ordre et l'estimation des paramètres ;
- la validation du modèle ;
- la prédiction.

Nous allons mettre en protique cette procédure dans ce TP en étudiant le jeu de données `airlinedata` (à télécharger depuis la page web des travaux pratiques) étudiée par Box and Jenkins (1976) et représentant le nombre de voyageurs en avion sur une période de 144 mois.

1 Analyse préliminaire

Graphe et première transformation
Avant de modéliser des données à l'aide d'un processus ARMA, il convient de représenter la série chronologique observée et de faire une première analyse de ses éventuelles tendances, saisonnalités ou autres particularités.

```
serie= scan("airlinedata.dat")
plot.ts(serie)
```

Son graphe indique une forte dépendance entre la variabilité de la série et son niveau : lorsque le temps augmente, les valeurs prises par la série sont de plus en plus étendues. Pour éliminer cette variabilité, on applique une transformation aux données observées. Ici le logarithme népérien :

```
lserie=log(serie)
plot.ts(lserie)
```

Sur le graphe du logarithme népérien de la série observée, on peut constater que cette transformation a stabilisé la variance des observations.

Analyse des autocorrélations
Il est assez simple de déceler une tendance ou bien une saisonnalité dans les données. Pour commencer, on s'intéresse à la structure de corrélation. On commence par représenter les nuages de points des observations (x_t) et (x_{t+h}) avec h entier, à l'aide de la commande `lag.plot` :

```
lag.plot(serie,lags=9,do.lines=F)
```

Par défaut, les nuages de point sont représentés pour h allant de 1 à *lags*. Il est possible de sélectionner certains nuages de points à l'aide de l'option `set.lags`. Par exemple, si on ne souhaite représenter que les nuages de points pour $h = 1$ et $h = 9$, il suffit d'appliquer la commande :

```
lag.plot(serie,set.lags=c(1,9),do.lines=F)
```

On peut aussi représenter la fonction d'autocorrélations empirique et la fonction d'autocorrélations

partielles empirique comme nous l'avons vu dans le TP précédent :

```
par(mfrow=c(2,1))
acf(serie,lag.max=50,plot=T,main="Fonction d'autocorrélation",
+xlab="Décalage")
pacf(serie,lag.max=50,plot=T,main="Fonction d'autocorrélation partielle",
+xlab="Décalage")
```

Rappelons qu'une tendance entraîne une persistance dans la fonction d'autocorrélation et une autocorrélation pratiquement périodique est un indice pour une composante saisonnière. Dans cet exemple, on observe une fonction d'autocorrélation empirique typique d'une série présentant une tendance et une composante saisonnière.

Tracez les mêmes graphiques avec la série transformée. Que constatez-vous ?

Supression de la tendance et de la saisonnalité
Si la série semble ne pas présenter les signes de la stationnarité, on peut alors essayer de la transformer. Nous avons vu dans le cours que la tendance et la saisonnalité pouvaient être éliminées en appliquant les opérateurs de différence ∇ et ∇_d à la série observée. En écrivant

```
mserie=diff(lserie)
plot.ts(mserie)
```

on applique l'opérateur de différence ∇ à `serie` et on élimine ainsi la tendance. L'objet retourné est de type `ts` et possède une observation de moins que `serie`. A vérifier.

Remarque : Pour supprimer la tendance, on pourrait également l'estimer et l'enlever directement des observations. Pour ce faire, on peut calculer la moyenne empirique

`mean()`

ou bien un estimateur de moindres carrés généralisé ou encore utiliser l'estimateur non paramétrique à moyenne mobile.

Pour enlever une saisonnalité d'ordre d, l'ordre de la différence doit être spécifié en indice par l'argument `lag`. La fonction `diff` calcule la série $x_{t+d} - x_t$ où d est donné par `lag`. Par exemple,

```
dserie=diff(mserie,lag=12)
plot.ts(dserie)
```

retourne la série ∇_d`(mserie)` et possède d observations en moins que `mserie`. En pratique, on pourra utiliser ces deux opérateurs pour enlever une tendance ou une saisonalité en "devinant" la valeur de d. La fréquence de la série analysée est une indication pour trouver la valeur de d, ainsi que le comportement de la fonction d'autocorrélation.

2 Estimation des paramètres de l'ARMA

Estimez maintenant les paramètres de l'ARMA de l'ordre que vous avez choisi.

3 Test sur les résidus

Validez maintenant le modèle.

4 Prévision

Proposez maintenant les 10 prochaines prédictions.

5 Validation avec R

Quel est le modèle proposé par R lorsqu'on ne lui spécifie pas les ordres ?
Correspond-il à celui que vous avez proposé ?

6 Deux applications

Récupérer le fichier de données `serie1.dat`.

1. Ce processus vous semble-t-il modélisable par un processus ARMA ? Pourquoi ?
2. On travaille désormais avec la série obtenue en appliquant la fonction diff à la série. Quelle transformation a-t-on effectuée ? Pourquoi ?
3. En observant les autocorrélations empiriques et autocorrélations partielles empiriques, proposer des modèles AR(p) et MA(q) d'ordre faible pour modéliser cette série.
4. Estimer les paramètres des deux modèles sélectionnés.
5. Tester la blancheur des résidus.
6. Conclure pour choisir un modèle.
7. S'inspirer de la démarche précédente pour modéliser la série `serie2.dat`.

M1 ISMAG
MIS243Y - Séries chronologiques
TP 3 - Evolution du PNB américain

On s'intéresse dans ce TP à la série du PNB américain (son logarithme plus précisement noté (X_t) dans la suite) sur la période 1954-1987 en dollars.
Commencer par charger le package "tseries".

```
library(tseries)
data(USeconomic)
PNB = USeconomic[, 2]
annee = seq(1954, 1987.75, 0.25)
plot(annee, PNB, main = "log(PNB) au cours du temps", t = "l",
+ col = "blue", xlab = "temps", ylab = "log PNB")
```

On constate clairement une tendance ; la série $\log(PNB)$ n 'est pas stationnaire.

1 Stationnarité de la série $\log(PNB)$

Traçons le corrélogramme et le corrélogramme partiel.

```
par(mfrow = c(1, 2))
acf(PNB)
pacf(PNB)
```

L'analyse de la fonction d'autocorrélation confirme la non stationnarité de la série $\log(PNB)$: en effet, la fonction d'autocorrélation empirique $\rho(h)$ n'est pas nulle pour tous les termes $h \geq 0$. Le test du portmanteau (testant la non-corrélation de l'échantillon) rejette aussi la stationnarité de la série :

```
Box.test(PNB)
```

2 Etude de la série $diff(\log(PNB))$

Différenciation, stationnarité et choix de modèle
Nous allons maintenant travailler sur la série $(Y_t = X_t - X_{t-1})_{2 \leq t \leq T}$. Sur le tracé de Y_t, on constate que la série ne semble plus avoir de tendance (comportement très erratique).

```
diffPNB = diff(PNB)
T = length(annee)
par(mfrow = c(1, 1))
plot(annee[2 :T], diffPNB, main = "diffPNB", t = "l",
col = "blue",xlab = "temps", ylab = "diff PNB")
```

On peut aussi tracer les fonctions d'autocorrélation empiriques.

```
par(mfrow = c(1, 2))
acf(diffPNB, lag.max = 30)
pacf(diffPNB, lag.max = 30)
```

Comme les fonctions d'autocorrélations (normale et partielle) décroissent très vite, on peut considérer que la série Y_t des différences est stationnaire. De plus, la série Y_t n'est pas centrée

malgré une faible moyenne empirique de 0.007596586. Ainsi, nous avons besoin d'un terme constant dans la modélisation de Y. En effet, lorsqu'on teste l'hypothèse "$\overline{Y}_T = 0$" avec le test de Student, on rejette l'hypothèse nulle.

```
mean(diffPNB)
t.test(diffPNB)
```

Pour la série X_t des log du PNB, cela signifie qu'elle est du type marche aléatoire puisque $Y_t = X_t - X_{t-1}$ est stationnaire. Au vu des autocorrélogrammes, un modèle ARMA(0,2) semble être le plus adapté. On peut aussi choisir les modèles ARMA(1,2) ou ARMA(8,2) ou encore ARMA(0,1). En effet, la fonction d'autocorrélation $\rho(h)$ est en dehors de l'intervalle de confiance pour $h = 1$ et $h = 2$ et la fonction d'autocorrélation partielle $\tau(h)$ est en dehors de l'intervalle de confiance pour $h = 1$ et quasiment en dehors de l'intervalle pour $h = 8$.

Estimation des modèles
L'estimation par maximum de vraisemblance se fait avec la fonction `arima` de R.

```
arima(diffPNB, c(0, 0, 2))
```

Estimons aussi les paramètres des trois autres modèles retenus :

```
arima(diffPNB, c(1, 0, 2))
arima(diffPNB, c(8, 0, 2))
arima(diffPNB, c(0, 0, 1))
```

Stationnarité des résidus
Pour comparer les modèles entre eux, on cherche à minimiser le critère AIC (Aikake's Information Criterion). Le modèle, dont le critère AIC est le plus bas (-863.73), est le modèle ARMA(0,2). Il semble donc être le mieux adapté aux données. Cependant, on peut aussi comparer ces 4 modèles en testant la stationnarité de leurs résidus.

```
residus02 = arima(diffPNB, c(0, 0, 2)))$residual
Box.test(residus02)
residus12 = arima(diffPNB, c(1, 0, 2)))$residual
Box.test(residus12)
residus82 = arima(diffPNB, c(8, 0, 2)))$residual
Box.test(residus82)
residus01 = arima(diffPNB, c(0, 0, 1)))$residual
Box.test(residus01)
```

On constate que le modèle ARMA(1,2) semble meilleur, puisque la p-valeur du test de Box Pierce est plus élevée. Ainsi, on ne peut pas rejeter le fait que les résidus sont non corrélés. Le tracé des autocorrélogrammes confirme que les modèles ARMA(1,2), ARMA(8,2) et ARMA(0,2) collent bien aux données.

```
par(mfrow = c(4, 2))
acf(residus02, main = "modèle ARMA(0,2)")
pacf(residus02, main = "modèle ARMA(0,2)")
acf(residus12, main = "modèle ARMA(1,2)")
pacf(residus12, main = "modèle ARMA(0,2)")
acf(residus82, main = "modèle ARMA(8,2)")
pacf(residus82, main = "modèle ARMA(0,2)")
acf(residus01, main = "modèle ARMA(0,1)")
pacf(residus01, main = "modèle ARMA(0,2)")
```

Normalité des résidus

On teste maintenant la normalité des résidus.

```
shapiro.test(residus02)
shapiro.test(residus12)
shapiro.test(residus82)
shapiro.test(residus01)
```

Seul le modèle ARMA(0,1) rejette la normalité des résidus à 95%.

Prévision

Nous allons maintenant tester les performances des 4 modèles. Pour ce faire, nous enlevons les 10 derniers points de notre série Y_t, pour ensuite comparer nos 10 valeurs prédites avec nos 10 valeurs réelles.

```
n = 10
index = 1 :(T - n - 1)
pred02 = predict(arima(diffPNB[index], c(0, 0, 2)), n)
```

```
par(mfrow = c(1, 3))
plot(annee[(T - 4 * n) :T], diffPNB[(T - 4 * n) :T - 1], main =
+ "prevision ARMA(0,2)", t = "l", col = "blue", xlab = "temps", ylab = "diff PNB")
lines(annee[(T - n) :T], c(diffPNB[T - n - 1], pred02$pred))
lines(annee[(T - n) :T], c(diffPNB[T - n - 1], pred02$pred) + c(0,pred02$se)
+ * 1.96, lty = 2)
lines(annee[(T - n) :T], c(diffPNB[T - n - 1], pred02$pred) - c(0,pred02$se)
+ * 1.96, lty = 2)
```

Faire la même chose pour les 3 autres modèles.

Au vu des graphes, on en déduit que les prédictions du modèle ARMA(0,1) sont plus mauvaises que celles des deux autres modèles. Les intervalles de confiance (à 95%) sur les valeurs prédites supposent que les résidus sont gaussiens (ce qui semblent être vrai pour le modèle ARMA(1,2), ARMA(8,2) et ARMA(0,2), puisque on ne peut pas rejetter l'hypothèse de normalité).

En conclusion de l'analyse de la série différenciée Y_t, on est amené à préférer le modèle ARMA(0,2), car il minimise le critère AIC sans perdre la stationnarité des résidus et qu'il nécessite moins de paramètres.

3 Modèle ARIMA pour la série $\log(PNB)$

Différenciation d'ordre 2 et choix de modèle
La série Y_t des différences de la série des log du PNB nous semblait centrée, mais si on trace l'histogramme, on constate qu'elle n'est pas du tout symétrique.

```
hist(diffPNB)
```

Ainsi, on va différencier une deuxième fois la série des log du PNB : on pose $Z_t = \Delta^2 X_t (= \Delta Y_t)$. On constate que la série est plus centrée et surtout plus symétrique.

```
d2PNB = diff(diffPNB)
par(mfrow = c(1, 2))
plot(annee[3 :T],d2PNB,main="Z au cours du temps",t="l",
+ col="blue",xlab="temps",ylab="Z")
par(mfrow = c(1, 2))
hist(d2PNB)
mean(d2PNB)
```

Le test de Student révèle qu'on ne peut pas rejetter l'hypothèse que $\overline{Z}_T = 0$.

```
t.test(d2PNB)
```

Ainsi nous allons utiliser un modèle ARIMA. Comme pour les modèles ARMA, nous allons maintenant trouver les coefficients p et q du modèle puis tester la stationnarité des résidus et la performance de nos prédictions.

```
par(mfrow = c(1, 2))
acf(d2PNB, lag.max = 30)
pacf(d2PNB, lag.max = 30)
```

Le tracé des autocorrélogrammes révèle que $q = 1$ et $p = 5$ ou 9.

Estimation des modèles
Nous allons donc tester deux modèles ARIMA(5,2,1) et ARIMA(9,2,1). On estime les coefficients :

```
arima(PNB, c(5, 2, 1), inc = FALSE)
arima(PNB, c(9, 2, 1), inc = FALSE)
```

Selon le critère AIC, le modèle ARIMA(5,2,1) semble mieux coller aux données.

Stationnarité et normalité des résidus
Testons la stationnarité et la normalité des résidus.

```
residus521=arima(PNB, c(5, 2, 1), inc = FALSE)$residual
Box.test(residus521)
residus921=arima(PNB, c(9, 2, 1), inc = FALSE)$residual
Box.test(residus921)
shapiro.test(residus521)
shapiro.test(residus921)
```

On en déduit qu'on ne peut pas rejetter la stationnarité et la normalité des résidus pour les modèles ARIMA(5,2,1) et ARIMA(9,2,1). De plus, le modèle ARIMA(5,2,1) apparaît meilleur.

```
par(mfrow = c(1, 2))
acf(residus521)
acf(residus921)
pacf(residus521)
pacf(residus921)
```

Enfin le tracé des fonctions d'autocorrélogrammes ne montre pas de corrélation pour les résidus. Au final, le modèle ARIMA(5,2,1) semble meilleur que l'ARIMA(9,2,1) sur ces données.

Prévision
Enfin testons la performance des prédictions.

```
index = 1 :(T - n)
pred521 = predict(arima(GNP[index], c(5, 2, 1), inc = FALSE),n)
pred921 = predict(arima(GNP[index], c(9, 2, 1), inc = FALSE),n)


plot(annee[(T - 4 * n) :T], PNB[(T - 4 * n) :T], main = "prevision ARIMA(5,2,1)",
t = "l", col = "blue", xlab = "temps", ylab = "PNB")
lines(annee[(T - n) :T], c(PNB[T - n], pred521$pred))
lines(annee[(T - n) :T], c(PNB[T - n], pred521$pred) + c(0, pred521$se)
 * 1.96, lty = 2)
lines(annee[(T - n) :T], c(PNB[T - n], pred521$pred) - c(0, pred521$se)
 * 1.96, lty = 2)
```

Refaire la même chose pour l'autre modèle. Au vu des prévisions, on conclut que le modèle ARIMA(5,2,1) est mieux adapté aux données.

4 Conclusion

Les tableaux ci-dessous récapitulent l'ensemble des résultats des différents modèles de TP.

Critères-Modèles	ARMA(0,2)	ARMA(0,1)	ARMA(1,2)	ARMA(8,2)
AIC	-863,73	-860,05	-861,89	-855,32
Vraisemblance	435,86	433,03	439,62	439,66
p-valeur Box Pierce	0,9474	0,5948	1	0,9997
p-valeur Shapiro	0,323	0,2492	0,4097	0,1872

FIGURE 1 – Bilan : critère AIC, log vraisemblance et tests sur les résidus pour les ARMA

Critères-Modèles	ARIMA(5,2,1)	ARIMA(9,2,1)
AIC	-847,31	-843,02
Vraisemblance	430,66	432,51
p-valeur Box Pierce	0,94	0,9561
p-valeur Shapiro	0,2780	0,2963

FIGURE 2 – Bilan : critère AIC, log vraisemblance et tests sur les résidus pour les ARIMA

En conclusion, le modèle ARMA(0,2) sur la série différenciée des PNB est meilleur compte tenu des critères choisis que le modèle ARIMA(5,2,1) sur la série PNB.

M1 ISMAG
MIS243Y - Séries chronologiques
TP 4 - Evolution du trafic SNCF

On s'intéresse dans ce TP à la série mensuelle du nombre de voyageurs kilomètres, correspondant au trafic deuxième classe du réseau principal SNCF pour la période 1963-1980 (source : Gouriéroux et Monfort, 1983).

```
Y = scan('trafic.dat')
Y = as.ts(Y)
Y = ts(Y,start=1963,frequency=12)
anneeY = seq(1963, 1980+11/12, by=1/12)
TY=length(anneeY)
```

On enlève à Y les 12 dernières valeurs (soit l'année 1980) afin de faire de l'estimation sur les années 1963 à 1979 puis comparer les prévisions issu du modèle estimé aux vraies valeurs.

```
n=12
index=1 :(TY-n)
annee=anneeY[index]
T=length(annee)
X=Y[index]
X = as.ts(X)
X = ts(X,start=1963,frequency=12)
anneeY = seq(1963, 1980+11/12, by=1/12)
plot(annee, X, main = "Evolution du trafic SNCF au cours du temps (1963-1980)",
+ t = "l", col = "blue", xlab = "temps", ylab = "nb voyageurs")
```

L'examen du graphe fait apparaître une tendance approximativement linéaire ; il sera donc nécessaire de différencier la série brute. La série présente visiblement une périodicité de période 12 qui pourra être prise en compte par l'introduction de polynômes retards B^{12}. Finalement, on peut remarquer qu'il n'y a pas d'évolution significative de la variabilité et il n'y a pas lieu a priori d'effectuer une transformation de type Box-Cox ou d'utiliser un modèle ARCH ou GARCH.

1 Stationnarité de la série

Traçons le corrélogramme et le corrélogramme partiel de X.

```
par(mfrow = c(1, 2))
acf(X)
pacf(X)
```

La fonction d'autocorrélation estimée $\hat{\rho}(h)$ est strictement positive pour les premières valeurs de h (au moins pour $h \leq 12$). Il y a donc lieu de différencier la série au moins une fois. Le test du portmanteau (testant la non-corrélation de l'échantillon) rejette aussi la stationnarité de la série :

```
Box.test(X)
```

2 Etude des séries $diff(X)$ et $diff12(diff(X))$

Nous allons maintenant travailler sur la série $(D_t = X_t - X_{t-1})_{2 \leq t \leq T}$. Sur le tracé de D_t, on constate que la série ne semble plus avoir de tendance (comportement très erratique).

```
diffX = diff(X)
T = length(annee)
par(mfrow = c(1, 1))
plot(annee[2 :T], diffX, main = "diffX", t = "l",
col = "blue",xlab = "temps", ylab = "diff X")
```

On peut aussi tracer les fonctions d'autocorrélation empiriques. Le corrélogramme estimé de la série différenciée $(I - B)X$ montre de fortes corrélations pour les valeurs h multiples de 12. Ceci implique qu'il faut appliquer au moins une fois l'opérateur $(I - B^{12})$ à la série des différences.

```
par(mfrow = c(1, 2))
acf(diffX, lag.max = 30)
pacf(diffX, lag.max = 30)
diff12X = diff(diff(X),lag=12)
par(mfrow = c(1, 1))
plot(annee[14 :T], diff12X, main = "Série Z", t = "l",
col = "blue",xlab = "temps", ylab = "Z")
par(mfrow = c(1, 2))
acf(diff12X, lag.max = 30)
pacf(diff12X, lag.max = 30)
```

Le corrélogramme de la série $Z_t = (I - B)(I - B^{12})X_t$ ne présente plus systématiquement de fortes valeurs des autocorrélations pour les petites valeurs de h ou pour les valeurs de h multiples de 12. On peut donc considérer que la série Z a été générée par un processus stationnaire. Il existe cependant de fortes corrélations pour $h = 1$ et $h = 12$. Ceci suggère d'introduire dans le polynôme moyenne mobile un terme du type $(I - \theta_1 B)(I - \theta_2 B^{12})$. Le choix d'une représentation moyenne mobile est confirmé par ma forme de la fonction d'autocorrélation empirique estimée. On choisit donc de modéliser le processus par un $SARIMA(0, 1, 1) \times (0, 1, 1)_{12}$.

Remarque :
(1) Le choix d'une représentation de type autorégressif aurait été possible aussi mais moins judicieux car il nécessiterait d'introduire un plus grand nombre de paramètres (puisqu'en particulier les sept premières valeurs de l'autocorrélation partielle estimée sont différentes de 0).
(2) Une moyenne mobile du type :

$$(I - \theta_1 B)(I - \theta_2 B^{12})\epsilon_t = \epsilon_t - \theta_1 \epsilon_{t-11} - \theta_2 \epsilon_{t-12} + \theta_1 \theta_2 \epsilon_{t-13}$$

admet des corrélations ρ_k non nulles pour $h = 1, 11, 12, 13$. Ceci est en particulier compatible avec les valeurs prises par $\hat{\rho}(11)$ et $\hat{\rho}(13)$.

Finalement, la moyenne empirique de Z est égale à -0.16 et la variance empirique vaut 28527. La variance de la moyenne empirique dans le cas d'une moyenne mobile du type $(I - \theta_1 B)(I - \theta_2 B^{12})\epsilon_t$ est asymptotiquement donnée par :

$$\frac{1}{T}\gamma(0)\left[1 + 2\rho(1) + 2\rho(11) + 2\rho(12) + 2\rho(13)\right]$$

où T désigne le nombre d'observations de $(I-\theta_1 B)(I-\theta_2 B^{12})\epsilon_t$. Cette variance est estimée par :

$$\frac{1}{204-13}28257\,[1+2(-0.4)+2(0.18)+2(-0.39)+2(0.18)]=21$$

La moyenne empirique est donc en module inférieur à deux fois son écart-type estimé et le processus peut être considéré comme centré. Ce qui est confirmé par le test de Student.

```
mean(diff12X)
t.test(diff12X)
```

3 Estimation du modèle

Le modèle retenu est un $SARIMA(0,1,1)\times(0,1,1)_{12}$:

$$(I-B)(I-B^{12})X_t=(I-\theta_1 B)(I-\theta_2 B^{12})\epsilon_t$$

avec $\mathbb{E}(\epsilon_t)=0$ et $\mathrm{Var}\epsilon_t)=\sigma^2$. Il y a donc trois paramètres à estimer : θ_1, θ_2 et σ^2. L'estimation par maximum de vraisemblance se fait avec la fonction `arima` de R.

```
arima(X,order=c(0,1,1),seasonal=list(order=c(0,1,1),period=12))
```

On obtient ainsi les estimées des trois paramètres ainsi que les écart-type estimés.

	θ_1	θ_2
Estimation	0.87	0.54
Ecart-type	0.04	0.07

La variance du bruit blanc est estimée par 26247.

4 Etude des résidus

Tout d'abord, on examine la suite des résidus d'estimation pour voir si elle ne présente pas certaines tendances ou irrégularités qui feraient rejeter l'hypothèse de bruit blanc. La série semble compatible avec cette hypothèse.

```
residus=arima(X,order=c(0,1,1),seasonal=list(order=c(0,1,1),period=12))$residual
par(mfrow=c(1,1))
plot(residus,main="Résidus au cours du temps",t="l",
+ col="blue",xlab="temps",ylab="résidus")
```

Stationnarité des résidus
Pour tester la stationnarité des résidus, on fait le test de Portmanteau et on trace les autocorrélogrammes.

```
Box.test(residus
par(mfrow=c(1,2))
acf(residus,main= "modele SARIMA(0,1,1)x(0,1,1)_1 2")
pacf(residus,main= "modele SARIMA(0,1,1)x(0,1,1)_1 2")
```

Normalité des résidus
Pour tester la normalité des résidus, on effecute le test de Shapiro-Wilks.

```
shapiro.test(residus)
```

5 Prévision

Nous allons maintenant tester les performances du modèle. Les valeurs prévues pour les 12 mois de l'année 1980 sont comparées aux valeurs réelles non prises en compte dans l'estimation.

```
pred = predict(arima(X,order=c(0,1,1),seasonal=list(order=c(0,1,1),period=12),12)
par(mfrow = c(1, 3))
plot(anneeY[(TY - 4 * n) :TY], Y[(TY - 4 * n) :TY - 1], main =
+ "prevision ARMA(0,2)", t = "l", col = "blue", xlab = "temps", ylab = "X")
lines(anneeY[(TY - n) :TY], c(X[T - n - 1], pred$pred),col="red")
lines(anneeY[(TY - n) :TY], c(X[T - n - 1], pred$pred) + c(0,pred$se)
+ * 1.96, lty = 2,col="green")
lines(anneeY[(TY - n) :TY], c(X[T - n - 1], pred$pred) - c(0,pred$se)
+ * 1.96, lty = 2,col="green")
```

M1 ISMAG
MIS243Y - Séries chronologiques
TP5 - Processus ARIMA/SARIMA et ARCH/GARCH

---

1 Les processus ARIMA et SARIMA

Simulation des processus ARIMA et SARIMA
Nous avons vu dans les TP précédents que la fonction `arima.sim(n,modele)` permet de simuler un processus ARMA(p,q) défini par

$$X_t - \sum_{k=1}^{p} a_k X_{t-k} = \epsilon_t - \sum_{j=1}^{q} b_j \epsilon_{t-j}.$$

Les paramètres a_k et bj du processus étant précisés dans le paramètre modele de la fonction :

```
modele=list(ar=c(a1, ... , ap),ma=c(b1, ... , bq))
```

Pour simuler un modèle ARIMA(p,d,q), il faut ajouter le paramètre `order=c(p,d,q)` dans le paramètre `modele` de la fonction `arima.sim` :

```
X=arima.sim(n,model=list(order = c(p,d,q),ar=c(a1, ... , ap),ma=c(b1, ... , bq)))
```

Pour simuler un modèle SARIMA$(p,d,q)\times(P,Q,D)_s$, il faut ajouter le paramètre `order=c(p,d,q)` et le paramètre `seasonal=list(order=c(P,D,Q),period=s)` dans le paramètre `modele` de la fonction `arima.sim` :

```
X=arima.sim(n,model=list(order = c(p,d,q),seasonal=list(order=c(P,D,Q),period=s),
  +ar=c(a1, ... , ap),ma=c(b1, ... , bq)))
```

Remarque : On peut aussi utiliser la syntaxe suivante pour un modèle ARIMA(p,d,q)

```
arima.sim(serie, model = list( ar = a, ndiff = 1,ma = theta), n = 2)
```

et pour un modèle SARIMA$(p,d,q)\times(P,D,Q)_s$, il faut utiliser une liste de deux éléments :

```
arima.sim(serie, model = list(list(ar = a, ndiff = d, ma = theta),
+ list(ar = A, ndiff = D, ma = Theta , period = s))
```

où `a`, `d`, `theta`, `A`, `D`, `Theta` et `s` sont les valeurs des coefficients du processus. Les fonctions précédentes peuvent alors être utilisée avec cet argument.

Estimation des processus ARIMA et SARIMA
Nous avons vu dans les TP précédents que la fonction `ar` permet d'estimer les paramètres d'un processus AR(p) :

```
X.ar=ar(X,aic=TRUE,order.max=NULL)
```

L'ordre p du processus autorégressif est choisi (inférieur à `order.max`) à l'aide du critère AIC (si l'option `aic` est validée).

Nous avons aussi vu que la fonction `arima` permet d'estimer un processus ARMA(p, q) :

```
X.arma=arima(X,order=c(p,0,q))
```

C'est aussi la fonction `arima` qui permet d'estimer les paramètres des processus ARIMA :

```
X.arima=arima(X,order=c(p,d,q))
```

et des processus SARIMA :

```
X.sarima=arima(X,order=c(p,d,q),seasonal=list(order=c(P,D,Q),period=T))
```

Parmi les sorties de cette fonction, on peut obtenir :

coef	estimation des coefficients des parties AR et MA, ainsi que de la constante
sigma2	estimation de la variance du bruit d'innovation
var.coef	estimation de la matrice de variance-covariance des paramètres
loglik	log-vraisemblance à l'optimum
aic	valeur du critère AIC (si la méthode du maximum de vraisemblance est appliquée)
resiudals	estimation des résidus

Prédiction des processus ARIMA et SARIMA
La fonction `predict(X,h)` permet d'effectuer une prévision à l'horizon h. Parmi les sorties de cette fonction,

pred	prévisions
se	écart-type de l'erreur de prévision

Il n'existe pas de fonction prédéfinie pour calculer un intervalle de confiance sur les prévisions, mais cela peut être fait manuellement grâce à ces deux sorties de la fonction `predict`.

Exercice 1 : On se propose d'étudier un ARIMA(1,1,1).

1. Générer 100 réalisations d'un tel processus.
2. Estimer les paramètres.
3. Prévoir les 5 prochaines valeurs du processus

Exercice 2 : Précipitations mensuelles à San Fransisco entre 1932 et 1966
Récupérer la série `sanfran.dat`.

1. La série semble-t-elle stationnaire ? Si non, faites en sorte qu'elle le soit.
2. Proposer alors un AR(p) adapté. Valider votre modélisation en testant les résidus.

   Mettez votre idée de côté car nous vous proposons de tester maintenant directement sur la série initiale un AR(2) avec partie saisonnière (autrement dit un SARIMA$(2,0,0) \times (0,0,0)_{12}$). On utilise désormais les données jusqu'à la fin de l'année 1963.
3. Estimer le modèle SARIMA$(2,0,0) \times (0,0,0)_{12}$. Afficher et tester les résidus.
4. Prévoir les précipitations de 1964, 1965 et 1966 à partir de ce modèle. Superposer sur un graphique prévision et valeurs réelles.

5. Faire de même avec le modèle AR(p) séléctionné à la première question (ainsi qu'avec un lissage exponentiel de Holt-Winters, avec et sans composante saisonnière).
6. Quel est, graphiquement, le meilleur modèle pour prédire cette série ?
7. Comment aurait-on pu répondre à la question précédente de façon moins subjective ? Comparer les résultats à l'analyse graphique.

Exercice 3 : Taux d'intérêt au Royaume-Uni
Le fichier `UKinterestrates.dat` contient le "spread" des taux d'intérêts (différence entre taux d'intérêt à long terme et à court terme) pour le Royaume-Uni entre mars 1953 et décembre 1996. En vous inspirant des exercices précédents, proposer une modélisation de type ARMA, ARIMA ou SARIMA. Justifier bien votre démarche.

2 Les processus ARCH et GARCH

Pour utiliser les fonctions spécifiques à l'étude des modèles ARCH et GARCH, il faut avant tout charger les packages `tseries` et `TSA`. Pour cela, aller dans le menu Packages puis cliquer sur Installer le(s) package(s)... puis sur France (Toulouse) dans la fenêtre CRAN Mirror et enfin sélectionner `tseries` et taper la commande `library(tseries)`. Refaire la même manipulation avec cette fois-ci le package `TSA`.

Exercice 3 : Données simulées
Soit le processus X_t défini par :

$$X_t|X_{t-1}, X_{t-2}, \ldots \sim \mathcal{N}(0, \sigma_t^2)$$

avec

$$\sigma_t^2 = 0.1 + 0.5X_{t-1}^2 + 0.2X_{t-2}^2$$

1. Reconnaissez-vous ce processus ?
2. Simuler dans un vecteur x 1000 réalisations de ce processus en tapant les commandes :

```
n =1100
a =c(0.1, 0.5, 0.2)                                   coefficients ARCH(2)
e =rnorm(n)
x =double(n)
x[1 :2] =rnorm(2, sd = sqrt(a[1]/(1.0-a[2]-a[3])))
for(i in 3 :n)                                        génération du processus
{x[i] = e[i]*sqrt(a[1]+a[2]*x[i-1]^ 2+a[3]*x[i-2]^ 2)}
x = ts(x[101 :1100])
```

   Représenter cette trajectoire graphiquement, et analyser les moyenne, variance et auto-covariance empiriques.
3. Ajuster un processus ARCH(p) au vecteur x. Contrôle (on pourra utilisr la commande `summary`) et conclusion.

La fonction `garch.sim` permet directement de simuler un processus GARCH. Par exemple, pour simuler le processus

$$X_t = \epsilon_t$$

avec $\epsilon_t|X_{t-1}, X_{t-2}, \ldots \sim \mathcal{N}(0, \sigma_t^2)$ et $\sigma_t^2 = 0.1 + 0.8X_{t-1}^2$, il faut écrire

```
monarch=garch.sim(alpha=c(0.1,0.8), n=500)
plot.ts(monarch)
```

La syntaxe est identique si une partie GARCH doit être modélisée. Par exemple, pour spécifier un modèle GARCH dont la variance conditionnelle est

$$\sigma_t^2 = 0.1 + 0.5X_{t-1}^2 + 0.3\sigma_{t-1}^2$$

on notera

```
monarch=garch.sim(alpha=c(0.1,0.5), beta=0.3,n=500)
plot.ts(monarch)
```

La fonction `garch` permet d'estimer un GARCH(p, q) :

```
serie=garch(data,order=c(q,p))
```

Parmi les sorties de cette fonction : `coef`, `residuals, fitted.values`.

La prédiction se fait de la même façon que pour les modèles de type ARMA.

Exercice 4 : Données réelles : EuStockMarkets
Le fichier `EuStockMarkets` de R contient les valeurs de cloture des 4 principaux indices boursiers européens, de 1991 à 1998. Pour chaque indice, chercher un modèle ARCH ou GARCH approprié, et effectuer la prévision à 30 jours. Les données pourront (devront ?) au préalable être transformées.

Exercice 5 : Données réelles : NYSE
Le fichier `nyse.dat` contient les évolutions journalières de la bourse des valeurs de New-York (NYSE) du 19 octobre 1984 au 31 décembre 1991. Chercher un processus ARCH ou GARCH adapté, et réaliser la prévision à 30 jours.