Séries temporelles – TD/TP 1
Premiers graphiques, saisonnalité, logarithme et résidus

M1 Actuariat – Université Gustave Eiffel

Durée indicative : 1h30

Objectifs

Cette séance accompagne la première heure et demie du chapitre 1. On travaille volontairement sur des séries qui ne sont pas celles du poly, afin de vérifier que les notions se transfèrent à de nouveaux exemples. Les deux séries utilisées sont fournies directement avec `R`, dans le package `datasets` : `JohnsonJohnson`, bénéfices trimestriels par action de Johnson & Johnson de 1960 à 1980, et `sunspot.year`, nombre annuel moyen de taches solaires.

Préparation

Les données sont déjà dans `R`. Il suffit de les charger et de leur donner des noms plus courts.

```
data(JohnsonJohnson)
data(sunspot.year)

jj <- JohnsonJohnson
sun <- sunspot.year
```

Décodage du code R

- `data(JohnsonJohnson)` demande à `R` de rendre disponible le jeu de données appelé `JohnsonJohnson`. Il est déjà fourni avec `R`.
- `data(sunspot.year)` fait la même chose pour la série annuelle des taches solaires.
- `jj <- JohnsonJohnson` crée un nom plus court : dans la suite, `jj` désigne la même série que `JohnsonJohnson`.
- `sun <- sunspot.year` crée de même un nom plus court pour `sunspot.year`.

Exercice 1 – Lire deux chronogrammes

Un *chronogramme* représente les observations dans l'ordre du temps. C'est presque toujours le premier graphique à tracer : il permet de repérer les phénomènes évidents avant de parler de modèle.

1) Tracer les deux séries.

```
plot.ts(jj, xlab = "annee", ylab = "benefice par action",
        main = "JohnsonJohnson : benefices trimestriels")
plot.ts(sun, xlab = "annee", ylab = "nombre annuel moyen",
        main = "sunspot.year : taches solaires")
```

Décodage du code R

- `plot.ts(...)` trace une série temporelle. Ici, `plot.ts(jj, ...)` trace la série `jj` et `plot.ts(sun, ...)` trace la série `sun`.
- Le premier argument, par exemple `jj`, indique quelle série doit être représentée.
- `xlab = "annee"` fixe le nom affiché sous l'axe horizontal.
- `ylab = "benefice par action"` ou `ylab = "nombre annuel moyen"` fixe le nom affiché à côté de l'axe vertical.
- `main = "..."` fixe le titre du graphique.
- Les guillemets indiquent du texte. Par exemple, `"annee"` n'est pas un objet `R`, mais l'étiquette que l'on veut écrire sur le graphique.

2) Compléter le tableau suivant. Il ne s'agit pas de faire un test statistique, mais de décrire proprement ce que l'on voit.

Série	Tendance ?	Motif régulier ?	Variabilité changeante ?	Commentaire concret
`jj`				
`sun`				

3) Pour chaque série, écrire une phrase du type : "ce qui domine visuellement cette série est...".

Idée à retenir

Un chronogramme ne donne pas encore un modèle. Il sert à formuler les premières questions : y a-t-il une croissance de long terme ? un motif qui revient ? des périodes plus instables que d'autres ?

Exercice 2 – Saisonnalité trimestrielle

La série `jj` est trimestrielle. On peut donc comparer les observations du premier trimestre entre elles, puis celles du deuxième trimestre, etc. C'est l'équivalent d'un *month plot*, mais avec quatre saisons au lieu de douze mois.

1) Tracer le graphique suivant.

```
monthplot(jj, ylab = "benefice par action",
          main = "JohnsonJohnson : comparaison des trimestres")
```

Décodage du code R

- `monthplot(jj, ...)` regroupe les observations de `jj` selon leur position dans le cycle saisonnier.
- Comme `jj` est trimestrielle, `R` obtient quatre groupes : trimestre 1, trimestre 2, trimestre 3 et trimestre 4.
- Le graphique permet donc de comparer les trimestres entre eux, au lieu de seulement suivre la série dans le temps.
- `ylab = "benefice par action"` et `main = "..."` jouent le même rôle que dans `plot.ts()` : étiquette de l'axe vertical et titre du graphique.

2) Quels trimestres semblent généralement plus hauts ? Quels trimestres semblent plus bas ?

3) Ce graphique est-il plus adapté que le chronogramme pour comparer les trimestres ? Pourquoi ?

4) La saisonnalité trimestrielle est-elle de même amplitude au début et à la fin de la période ?

Exercice 3 – Pourquoi regarder le logarithme ?

Sur `jj`, l'amplitude des oscillations semble augmenter avec le niveau de la série. Cela suggère que les variations sont peut-être mieux décrites en valeur relative qu'en valeur absolue.

1) Tracer la série brute puis son logarithme.

```
plot.ts(jj, xlab = "annee", ylab = "benefice par action",
        main = "JohnsonJohnson")
plot.ts(log(jj), xlab = "annee", ylab = "log(benefice)",
        main = "JohnsonJohnson : logarithme")
```

Décodage du code R

- La première commande trace la série `jj` sur son échelle d'origine.
- `log(jj)` construit une nouvelle série : chaque observation de `jj` est remplacée par son logarithme.
- La deuxième commande trace donc le logarithme de la série, pas la série brute.
- Le reste de la syntaxe est identique : `xlab` règle l'axe horizontal, `ylab` règle l'axe vertical et `main` règle le titre.

2) Sur quelle échelle les oscillations sont-elles les plus comparables : l'échelle brute ou l'échelle logarithmique ?

3) Supposons que $x_t = m_t(1 + u_t)$, où m_t est le niveau de la série et u_t une variation relative. Calculer $\log(x_t)$. Que vaut approximativement $\log(1 + u_t)$ lorsque u_t est petit ?

4) Expliquer en une phrase pourquoi le logarithme est souvent utile pour une série économique positive qui croît fortement.

Exercice 4 – Valeurs ajustées et résidus

On ajuste maintenant une droite à $\log(\texttt{jj})$. Ce n'est pas encore un modèle de prévision : c'est une première manière de séparer une tendance simple et ce qui reste autour.

1) Ajuster une droite et la superposer au chronogramme de $\log(\texttt{jj})$.

```
temps <- time(jj)
temps.c <- temps - 1960
log.jj <- as.numeric(log(jj))
reg.jj <- lm(log.jj ~ temps.c)
res <- residuals(reg.jj)
```

Décodage du code R

- `time(jj)` récupère les dates associées aux observations de `jj`. Pour une série trimestrielle, `R` connaît aussi les fractions d'année.
- `temps <- time(jj)` stocke ces dates dans l'objet `temps`.
- `temps.c <- temps - 1960` recentre le temps : au lieu de travailler avec les années autour de 1960, on travaille avec le nombre d'années écoulées depuis 1960.
- `log.jj <- as.numeric(log(jj))` calcule le logarithme de `jj`, puis le transforme en simple vecteur numérique.
- `lm(log.jj ~ temps.c)` ajuste une régression linéaire : le symbole `~` sépare la variable à expliquer de la variable explicative. Ici, `log.jj` est expliqué par une droite en fonction de `temps.c`.
- `reg.jj <- ...` stocke le résultat de cette régression dans l'objet `reg.jj`.
- `residuals(reg.jj)` extrait les résidus de la régression, c'est-à-dire les écarts entre les observations et la droite ajustée.

```
plot(temps, log.jj, type = "l", xlab = "annee",
     ylab = "log(benefice)", main = "log(jj) et droite ajustee")
abline(a = coef(reg.jj)[1] - 1960 * coef(reg.jj)[2],
       b = coef(reg.jj)[2], lty = 2, lwd = 2)
```

Décodage du code R

- `plot(temps, log.jj, ...)` trace `log.jj` en fonction de `temps`. Le premier argument donne l'axe horizontal, le second l'axe vertical.
- `type = "l"` signifie que l'on veut relier les points par une ligne. La lettre `l` vient de *line*.
- `abline(...)` ajoute une droite sur un graphique déjà tracé.
- Dans `abline(a = ..., b = ...)`, `a` est l'ordonnée à l'origine et `b` est la pente de la droite.
- `coef(reg.jj)` donne les coefficients estimés de la régression. `coef(reg.jj)[1]` est l'interception, et `coef(reg.jj)[2]` est la pente associée à `temps.c`.
- `lty = 2` demande une ligne en pointillés. `lwd = 2` demande une ligne plus épaisse.

Quelle est ici la valeur ajustée $\widehat{x}_t$?

2) Tracer les résidus.

```
plot(temps, res, type = "l", xlab = "annee", ylab = "residu",
     main = "Residus apres tendance lineaire")
abline(h = 0, lty = 2)
```

Décodage du code R

- `plot(temps, res, type = "l", ...)` trace les résidus `res` dans l'ordre du temps.
- L'axe horizontal est encore `temps`. L'axe vertical est maintenant `res`, c'est-à-dire le reste après ajustement de la droite.
- `abline(h = 0, ...)` ajoute une droite horizontale au niveau 0. Cela aide à voir si les résidus sont plutôt positifs ou négatifs.
- `lty = 2` rend cette ligne horizontale pointillée.

Donner la formule du résidu empirique $\widehat{z}_t$.

3) Les résidus ressemblent-ils à un bruit complètement désorganisé autour de zéro ? Ou voit-on encore des motifs ?

4) Tracer le lag plot des résidus au retard 1.

```
n <- length(res)
plot(res[-n], res[-1],
     xlab = "residu a la date precedente",
     ylab = "residu a la date courante",
     main = "Lag plot des residus")
abline(h = 0, v = 0, lty = 2, col = "grey")
```

Décodage du code R

- `length(res)` donne le nombre de résidus. On le stocke dans `n`.
- `res[-n]` signifie : tous les résidus sauf le dernier.
- `res[-1]` signifie : tous les résidus sauf le premier.
- Le graphique met donc en abscisse un résidu et en ordonnée le résidu suivant. On compare ainsi $\widehat{z}_t$ et $\widehat{z}_{t+1}$.
- `abline(h = 0, v = 0, ...)` ajoute à la fois une ligne horizontale d'ordonnée 0 et une ligne verticale d'abscisse 0.
- `col = "grey"` demande que ces lignes soient grises.

Que suggère ce graphique sur les résidus consécutifs ?

Idée à retenir

Un résidu est un reste après ajustement. S'il contient encore une structure visible, cela signifie que le modèle très simple utilisé n'a pas capturé toute l'information temporelle.

Exercice 5 – Mini-synthèse

Dire si les phrases suivantes sont correctes ou incorrectes, puis corriger si nécessaire.

1) Une saisonnalité est un motif qui revient régulièrement.
2) Le logarithme est utile uniquement pour les séries mensuelles.
3) Un résidu est l'écart entre une observation et la valeur ajustée par le modèle.
4) Si les résidus montrent encore un motif clair, cela peut indiquer que le modèle ajusté est trop simple.