# Séries temporelles – TD/TP 2
Manipuler des objets `ts` dans `R`

M1 Actuariat – Université Gustave Eiffel

Durée indicative : 1h30

**Objectifs**

Cette séance accompagne le chapitre 2. L'objectif est de savoir contrôler l'information temporelle contenue dans un objet `ts` : début, fin, fréquence, dates numériques, cycles, sous-séries, différences et valeurs manquantes. Les données utilisées sont fournies avec `R`, dans le package `datasets` : `co2`, concentration mensuelle de CO2 à Mauna Loa, et `UKgas`, consommation trimestrielle de gaz au Royaume-Uni.

## Préparation

```
data(co2)
data(UKgas)

x <- co2
gas <- UKgas
```

**Décodage du code R**

- `data(co2)` rend disponible la série `co2`, déjà fournie avec `R`. Elle contient des observations mensuelles.
- `data(UKgas)` rend disponible la série `UKgas`, elle aussi fournie avec `R`. Elle contient des observations trimestrielles.
- `x <- co2` crée un nom plus court : dans la suite, `x` désigne la même série que `co2`.
- `gas <- UKgas` crée de même un nom plus court pour `UKgas`.
- Le symbole `<-` signifie "affecter" : on stocke un objet sous un nom pour pouvoir le réutiliser.

## Exercice 1 – Contrôler l'horloge d'une série

Avant de tracer ou de transformer une série, il faut vérifier comment `R` la situe dans le temps.

1) Pour `x` et `gas`, afficher la date de début, la date de fin et la fréquence.

```
start(x); end(x); frequency(x)
start(gas); end(gas); frequency(gas)
```

**Décodage du code R**

- `start(x)` affiche la première date associée à la série `x`.
- `end(x)` affiche la dernière date associée à la série `x`.
- `frequency(x)` affiche le nombre d'observations par unité de temps. Pour une série mensuelle, ce nombre vaut 12 ; pour une série trimestrielle, il vaut 4.
- Le point-virgule `;` permet d'écrire plusieurs commandes sur la même ligne. Ici, il sert seulement à gagner de la place.
- Les trois commandes suivantes font la même chose pour `gas`.

2) Compléter le tableau suivant.

| Série | Début | Fin | Fréquence | Interprétation |
|---|---|---|---|---|
| `co2` | | | | |
| `UKgas` | | | | |

3) Expliquer en une phrase ce que signifie `frequency(x) = 12`.

**Idée à retenir**

La fréquence de `R` n'est pas encore une fréquence spectrale. Elle indique seulement combien d'observations il y a dans une unité de temps, par exemple douze mois dans une année.

## Exercice 2 – Chronogramme et zoom avec `window()`

1) Tracer le chronogramme de `x`.

```
plot.ts(x, xlab = "annee", ylab = "CO2 (ppm)",
        main = "Concentration mensuelle de CO2")
```

**Décodage du code R**

- `plot.ts(x, ...)` trace la série temporelle `x` dans l'ordre du temps.
- `xlab = "annee"` fixe le nom de l'axe horizontal.
- `ylab = "CO2 (ppm)"` fixe le nom de l'axe vertical ; `ppm` signifie "parties par million".
- `main = "..."` fixe le titre du graphique.
- Les informations de début, fin et fréquence sont déjà contenues dans l'objet `x`, donc il n'est pas nécessaire de construire soi-même l'axe des dates.

2) Extraire la sous-série de janvier 1990 à décembre 1992, puis la tracer.

```
x.9092 <- window(x, start = c(1990, 1), end = c(1992, 12))
plot.ts(x.9092, xlab = "annee", ylab = "CO2 (ppm)",
        main = "CO2 : zoom 1990-1992")
```

**Décodage du code R**

- `window(x, ...)` extrait une partie de la série `x`.
- `start = c(1990, 1)` signifie que l'on commence en 1990, période 1. Comme `x` est mensuelle, la période 1 est janvier.
- `end = c(1992, 12)` signifie que l'on termine en 1992, période 12, donc en décembre.
- `x.9092 <- ...` stocke cette sous-série dans un nouvel objet appelé `x.9092`.
- La commande `plot.ts(x.9092, ...)` trace ensuite uniquement cette sous-série.

3) Qu'est-ce que le zoom permet de voir plus facilement que le graphique complet ?

4) Vérifier la date de début, la date de fin et la fréquence de `x.9092`. Que constate-t-on ?

## Exercice 3 – Temps numérique et cycle annuel

1) Afficher les douze premières dates numériques et les douze premiers cycles de `x`.

```
head(time(x), 12)
head(cycle(x), 12)
```

**Décodage du code R**

- `time(x)` renvoie les dates numériques associées aux observations de `x`. Pour une série mensuelle, les mois apparaissent comme des fractions d'année.
- `cycle(x)` renvoie la position de chaque observation dans le cycle annuel : 1 pour janvier, 2 pour février, etc.
- `head(objet, 12)` affiche seulement les 12 premières valeurs de `objet`. C'est utile pour regarder un début de série sans tout imprimer.
- La première ligne affiche donc les 12 premières dates ; la deuxième ligne affiche les 12 premiers mois dans le cycle.

2) Que représente `time(x)` ? Que représente `cycle(x)` ?

3) À quoi peut servir `cycle(x)` pour étudier une série mensuelle ?

## Exercice 4 – Différence saisonnière

La série `co2` est mensuelle. La différence de retard 12 compare chaque mois au même mois de l'année précédente.

1) Calculer et tracer la différence saisonnière de retard 12.

```
dx <- diff(x, lag = 12)
plot.ts(dx, xlab = "annee", ylab = "difference annuelle",
        main = "CO2 : difference saisonniere")
abline(h = 0, lty = 2)
```

**Décodage du code R**

- `diff(x, lag = 12)` calcule les différences $x_t - x_{t-12}$. Comme `x` est mensuelle, cela compare un mois au même mois de l'année précédente.
- `dx <- ...` stocke cette nouvelle série dans l'objet `dx`.
- `plot.ts(dx, ...)` trace la série des différences annuelles.
- `abline(h = 0, lty = 2)` ajoute une ligne horizontale en 0, en pointillés. Cela aide à voir si les différences sont plutôt positives ou négatives.

2) Quelle grandeur la série `dx` représente-t-elle concrètement ?

3) La différence saisonnière supprime-t-elle complètement toute structure ? Justifier à partir du graphique.

## Exercice 5 – Valeurs manquantes

On crée volontairement une courte période manquante pour voir comment `R` la traite.

1) Créer une copie de `x`, puis remplacer par `NA` les observations d'avril à septembre 1991.

```
x.miss <- x
t <- time(x.miss)
ind <- t >= 1991 + 3/12 & t <= 1991 + 8/12
x.miss[ind] <- NA
```

**Décodage du code R**

- `x.miss <- x` crée une copie de `x`. On évite ainsi de modifier la série d'origine.
- `t <- time(x.miss)` récupère les dates numériques de cette copie.
- `ind <- ...` crée un vecteur logique : pour chaque date, il vaut `TRUE` si la date est comprise entre avril et septembre 1991, et `FALSE` sinon.
- `&` signifie "et" : les deux conditions doivent être vraies en même temps.
- `x.miss[ind] <- NA` remplace par `NA` les observations pour lesquelles `ind` vaut `TRUE`. En `R`, `NA` signifie "valeur manquante".

2) Vérifier le nombre de valeurs manquantes, puis tracer le zoom 1990–1992.

```
sum(is.na(x.miss))
plot.ts(window(x.miss, start = c(1990,1), end = c(1992,12)),
        xlab = "annee", ylab = "CO2 (ppm)",
        main = "CO2 avec valeurs manquantes")
```

**Décodage du code R**

- `is.na(x.miss)` teste chaque observation : le résultat vaut `TRUE` si l'observation est manquante, `FALSE` sinon.
- `sum(is.na(x.miss))` compte le nombre de valeurs manquantes, car `R` additionne les `TRUE` comme des 1 et les `FALSE` comme des 0.
- `window(x.miss, start = c(1990,1), end = c(1992,12))` extrait le zoom 1990–1992.
- Le graphique montre une interruption : cela correspond aux observations remplacées par `NA`.

3) Pourquoi ne faut-il pas confondre une valeur manquante avec la valeur 0 ?

4) Proposer une interpolation linéaire pour remplacer ces valeurs manquantes. Cette interpolation est-elle une observation réelle ?