MUSÉUM NATIONAL D'HISTOIRE NATURELLE

SEMIN-R

Introduction à l'analyse de séries temporelles sour R

Jérôme SUEUR
UMR 5202 Origine Structure et Evolution de la Biodiversité
Département Systématique et Evolution, MNHN
sueur@mnhn.fr

Introduction aux séries temporelles sous R

Jérôme Sueur

MNHN – Systématique et Evolution
UMR CNRS 5202 – OSEB
sueur@mnhn.fr

21 Novembre 2008

Plan

1. Définition
2. Le choix dans la date
3. Objets ts
4. Manipulations
5. Analyses simples
6. Références

C'est quoi une série temporelle ?

Une série temporelle est une collection de données otenue de manière séquentielle au cours du temps.

Il y a donc typiquement deux variables associées :

- une variable quantitative dont les données sont dépendantes
- une variable "temps"

C'est quoi une série temporelle ?

Exemples :

- données météorologiques
- données financières
- phénomènes ondulatoires (son, signal neuronique, etc)
- suivi de populations
- ...

La date du jour

La fonction `date()` retourne la date du jour sous forme d'une chaîne de caractères :

```
> j <- date()
> j
```

```
[1] "Thu Nov 20 17:16:30 2008"
```

```
> class(j)
```

```
[1] "character"
```

La class Date

Il y a une classe spéciale pour les dates, c'est la classe `Date`.

La coercion character ⇒ Date se fait avec `as.date()`.

Attention à l'ordre des données !

```
> dates <- c("17/02/92", "27/02/92", "14/01/92", "28/02/92", "01/02/92",
+     "02/01/92")
> dates <- as.Date(dates, "%d/%m/%y")
> dates
```

```
[1] "1992-02-17" "1992-02-27" "1992-01-14" "1992-02-28" "1992-02-01"
[6] "1992-01-02"
```

Durée

Pour calculer une différence temporelle, utiliser `difftime()` avec un argument pour l'unité de temps :

```
> difftime(dates[1], dates[2])

Time difference of -10 days

> difftime(dates[1], dates[2], units = "s")

Time difference of -864000 secs
```

Série de dates

Il y a un équivalent à la fonction générique `seq`, il s'agit de `seq.Date`. Son utilisation est similaire. Elle fait appel aux mêmes arguments :

- `from` et `to` pour le début et la fin de la séquence
- `by` pour le pas temporel
- `length.out` pour la longueur de l'objet.

Exemple 1 : les 15 premiers jours du mois de novembre 2008 :

```
> d1 <- seq(from = as.Date("01/11/08", "%d/%m/%y"), to = as.Date("15/11/08",
+     "%d/%m/%y"), by = "day")
> d1
 [1] "2008-11-01" "2008-11-02" "2008-11-03" "2008-11-04" "2008-11-05"
 [6] "2008-11-06" "2008-11-07" "2008-11-08" "2008-11-09" "2008-11-10"
[11] "2008-11-11" "2008-11-12" "2008-11-13" "2008-11-14" "2008-11-15"
```

Série de dates

Exemple 2 : le 15 de chaque mois de 2009 :

```
> d2 <- seq(from = as.Date("15/01/09", "%d/%m/%y"), to = as.Date("15/12/09",
+     "%d/%m/%y"), by = "month")
> d2

 [1] "2009-01-15" "2009-02-15" "2009-03-15" "2009-04-15" "2009-05-15"
 [6] "2009-06-15" "2009-07-15" "2009-08-15" "2009-09-15" "2009-10-15"
[11] "2009-11-15" "2009-12-15"
```

Exemple 3 : un jour sur 2 au cours du mois de mars 2009 :

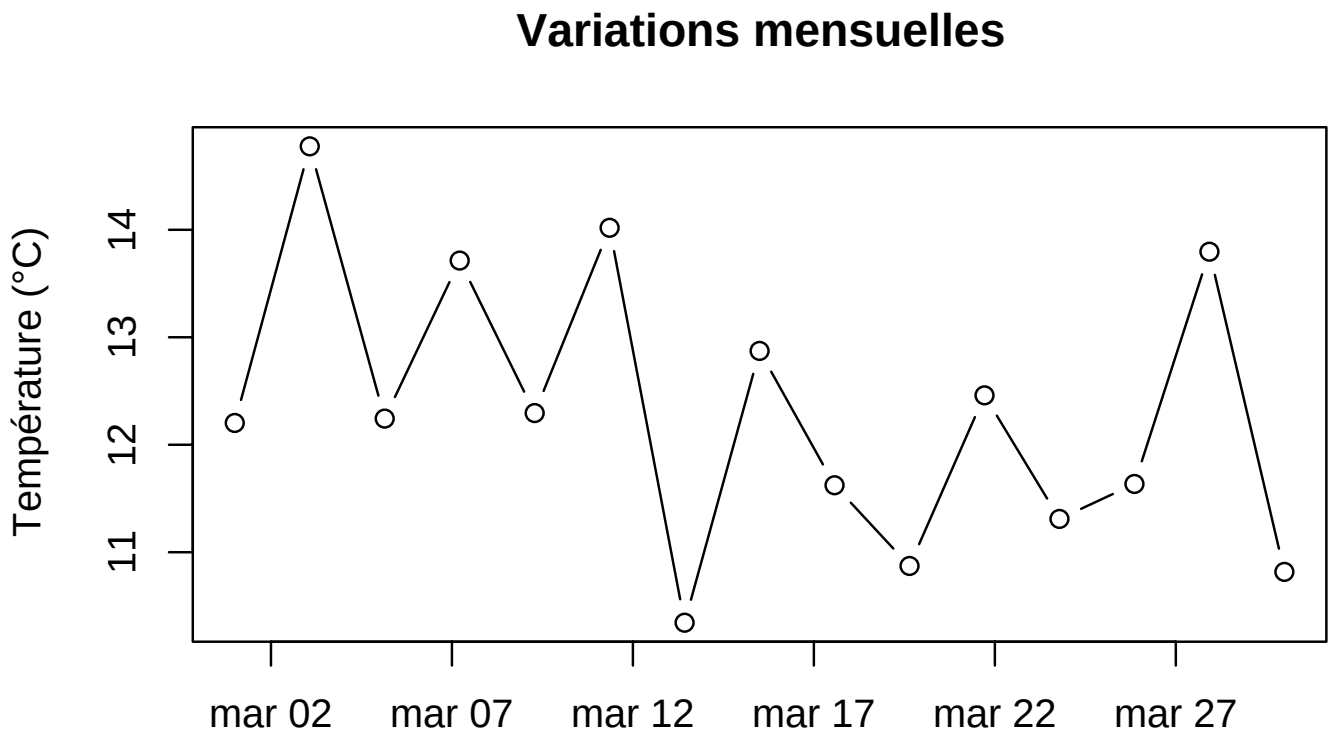
```
> d3 <- seq(from = as.Date("01/03/09", "%d/%m/%y"), to = as.Date("30/03/09",
+     "%d/%m/%y"), length.out = 15)
> d3

 [1] "2009-03-01" "2009-03-03" "2009-03-05" "2009-03-07" "2009-03-09"
 [6] "2009-03-11" "2009-03-13" "2009-03-15" "2009-03-17" "2009-03-19"
[11] "2009-03-21" "2009-03-23" "2009-03-25" "2009-03-27" "2009-03-30"
```

Série de dates

On peut alors faire des plots simples :

```
> r <- rnorm(length(d3), mean = 12)
> plot(x = d3, y = r, type = "b", ylab = "Température (°C)",
+     main = "Variations mensuelles")
```



Les objets "Séries temporelles simples"

La fonction `ts` permet de créer des séries temporelles.
Les fonctions corrolaires habituelles sont `as.ts` et `is.ts`.

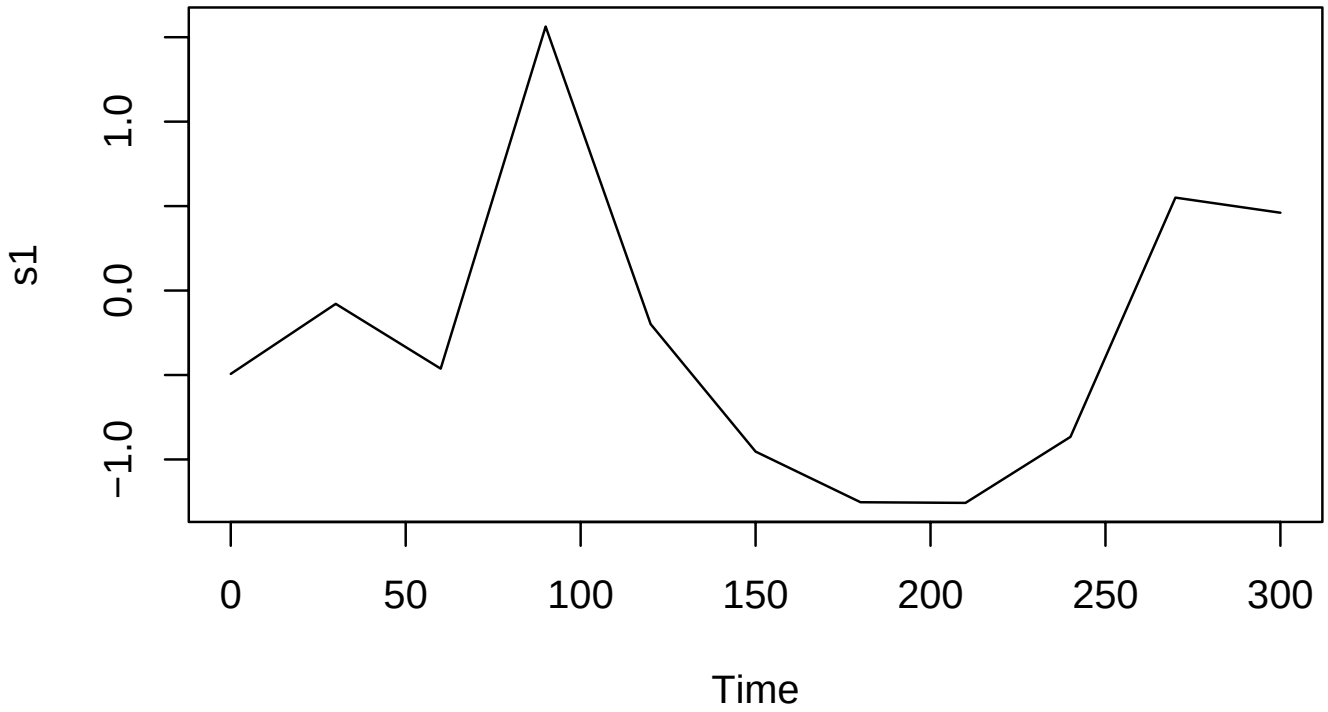
Il y a cinq arguments principaux :

- `data` : les données décrivant la série temporelle
- `start` : le temps de déart
- `end` : le temps de fin
- `frequency` : la fréquence d'échantillonnage ou le nombre d'observations par unité de temps
- `deltat` : la période entre deux observations successives

Les objets "Séries temporelles simples"

Exemple 1 : des observations faites toutes les 30 s pendant 5 minutes

```
> s1 <- ts(data = rnorm(11), start = 0, end = 5 * 60, frequency = 1/30)
> plot(s1)
```



Les objets "Séries temporelles simples"

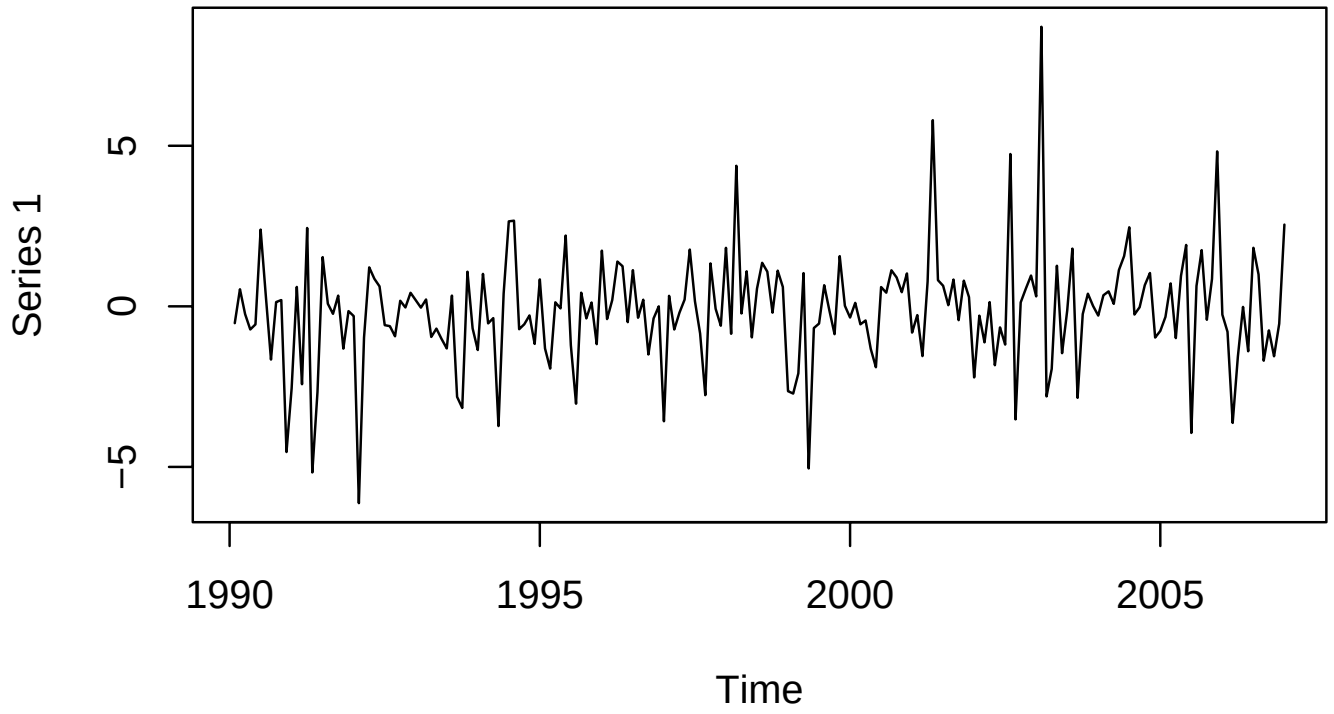
Exemple 2 : des observations mensuelles commençant en février 1990

```
> s2 <- ts(matrix(rt(204, df = 3)), start = c(1990, 2), frequency = 12)
> s2
```

	Jan	Feb	Mar	Apr	May
1990		-0.524636623	0.532366357	-0.242391629	-0.720353589
1991	-2.547032225	0.605345517	-2.424523557	2.440342424	-5.171540101
1992	-0.302709581	-6.127893431	-0.975534955	1.213668310	0.857832666
1993	0.190850312	-0.039613428	0.212922834	-0.952682567	-0.692679562
1994	-1.359153545	1.010396888	-0.530349724	-0.366999161	-3.727558147
1995	0.839360606	-1.311023737	-1.938757075	0.125474572	-0.063375544
1996	1.734610428	-0.396759989	0.194306752	1.394864517	1.251138489
1997	-3.578833421	0.327519648	-0.721123914	-0.198914479	0.208983737
1998	1.822099811	-0.854731310	4.377070862	-0.221107607	1.090560494
1999	-2.641260485	-2.716353020	-2.087236483	1.035084046	-5.047291299
2000	-0.347719049	0.108180056	-0.560152219	-0.439479151	-1.325060766
2001	-0.817357166	-0.272129418	-1.549500391	0.771997527	5.797192474
2002	-2.213424503	-0.288842699	-1.121214726	0.131623594	-1.832936020
2003	0.310262267	8.705558534	-2.804783910	-1.948258344	1.264606859
2004	-0.286865930	0.345136281	0.467093595	0.077903337	1.131445486
2005	-0.766372846	-0.327541704	0.714106155	-0.988566616	0.946688607
2006	-0.258164020	-0.793393561	-3.629717860	-1.567065286	-0.017383407

Les objets "Séries temporelles simples"

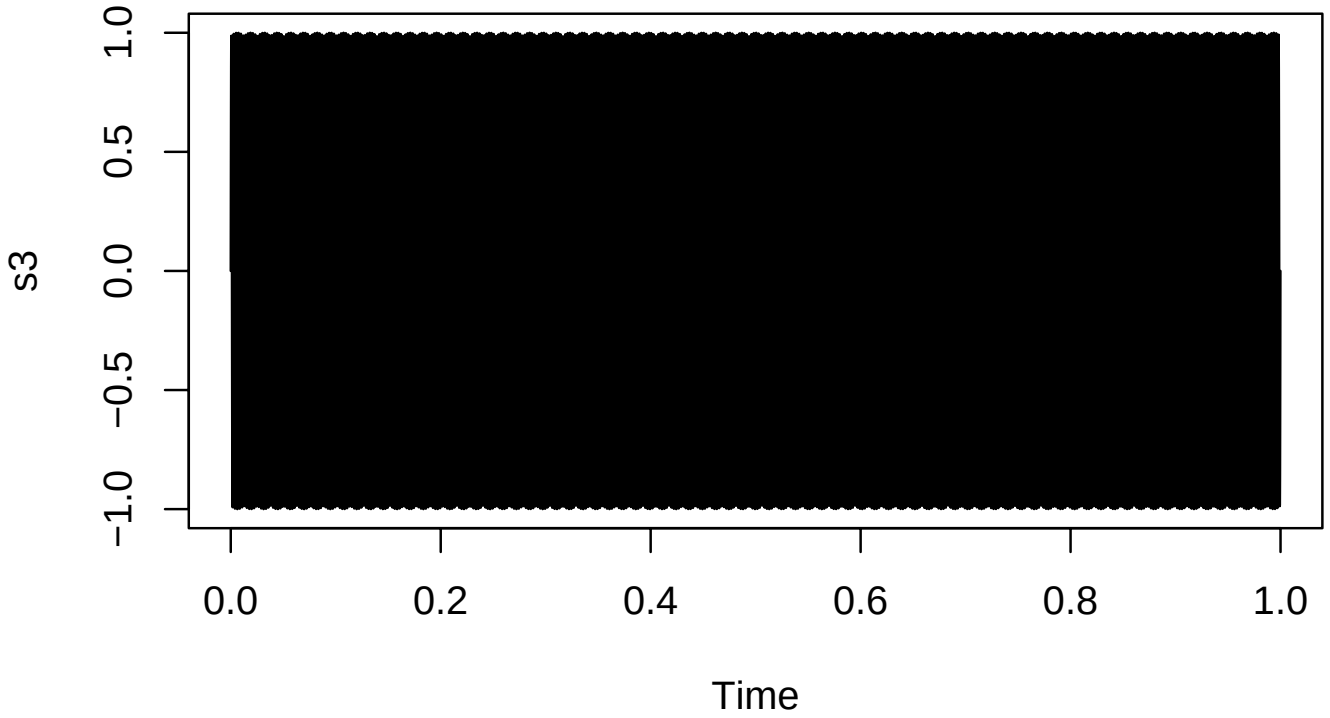
```
> plot(s2)
```



Les objets "Séries temporelles simples"

Exemple 3 : un son de 440 Hz durant 1s échantillonné à 8000 Hz

```
> s3 <- ts(sin(2 * pi * 440 * seq(0, 1, length.out = 8000)), start = 0,
+     frequency = 8000)
> plot(s3)
```



Les objets "Séries temporelles simples"

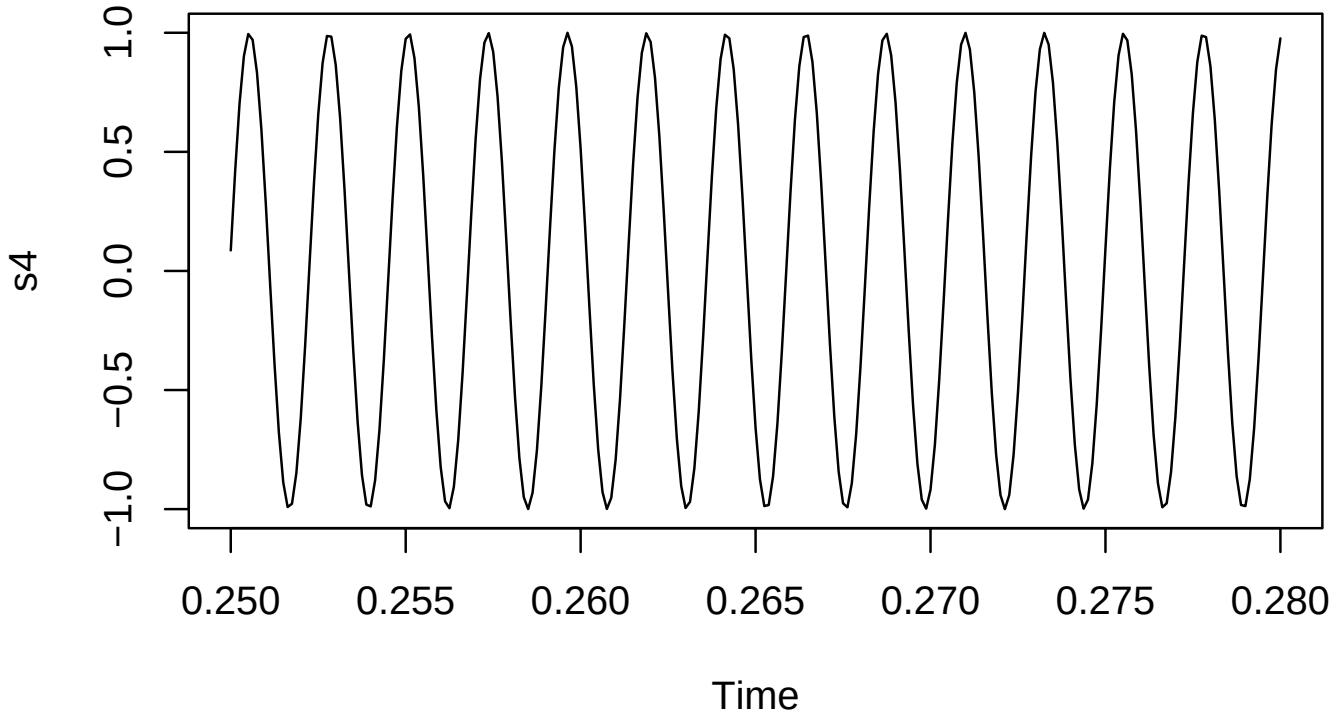
Pour avoir des informations concernant la série temporelle :

```
> start(s3)
[1] 0 1
> end(s3)
[1]    0 8000
> frequency(s3)
[1] 8000
> deltat(s3)
[1] 0.000125
```

Découpe

Pour couper une partie de la série, deux possibilités :

```
> s4 <- ts(s3, start = 0.25, end = 0.28, freq = 8000)
> s4 <- window(s3, start = 0.25, end = 0.28)
> plot(s4)
```



Les objets "séries temporelles multiples"

Les données de plusieurs séries sont combinées sous forme d'une matrice, chaque colonne correspondant à une série. La classe de l'objet est `mts` et `ts` :

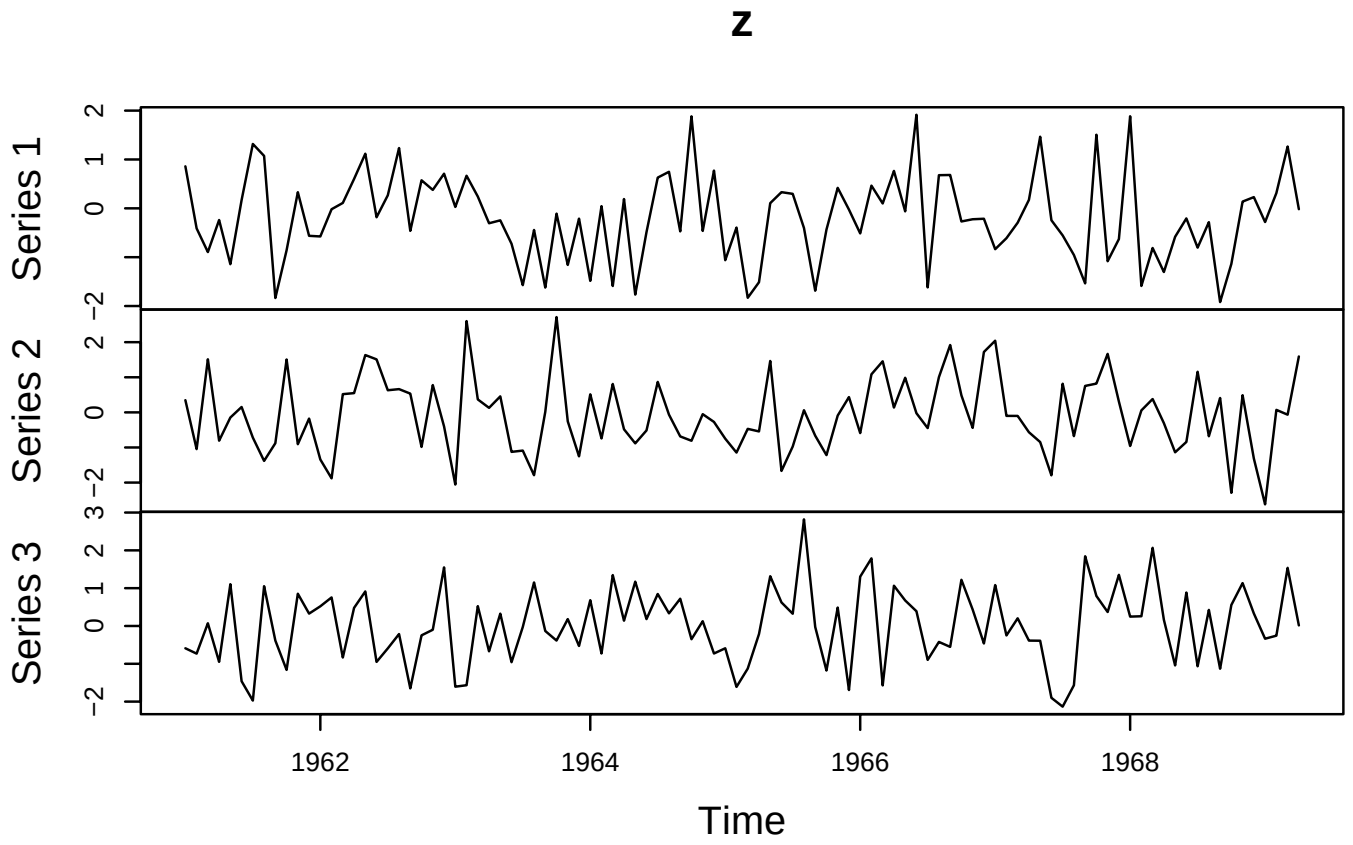
```
> z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), frequency = 12)
> class(z)
```

```
[1] "mts" "ts"
```

Les objets "séries temporelles multiples"

La fonction `plot` prend en charge `mts` en séparant les séries ou... :

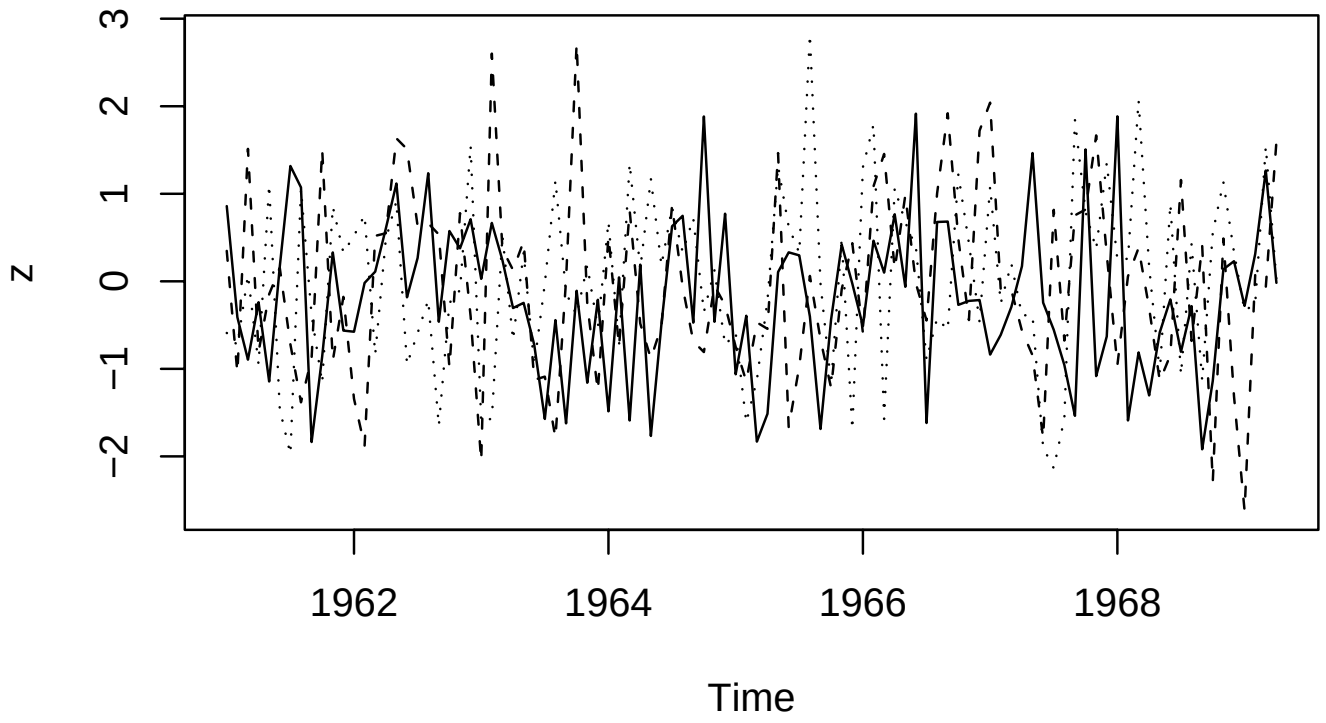
```
> plot(z)
```



Les objets "séries temporelles multiples"

... en les associant sur un unique graphique :

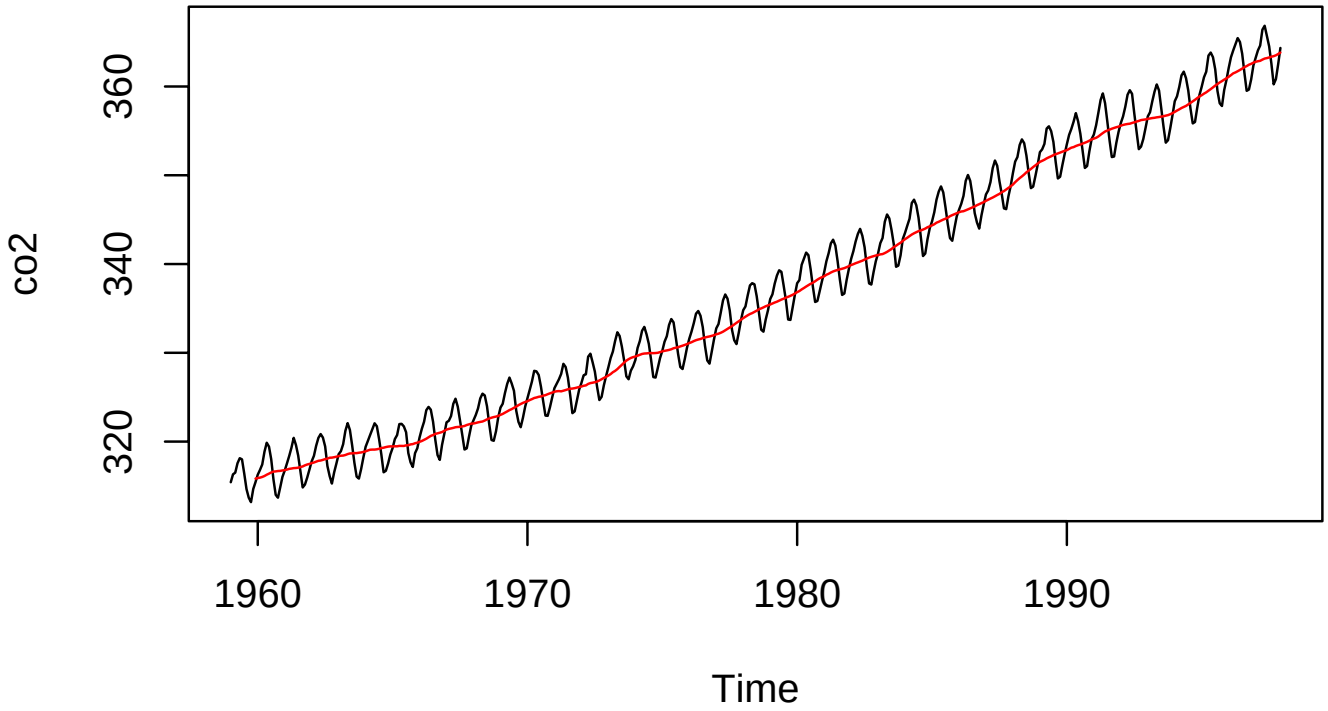
```
> plot(z, plot.type = "single", lty = 1:3)
```



Lissage : moyenne glissante

Moyenne glissante par convolution avec la fonction `filter` :

```
> data(co2)
> plot(co2, ylab = "co2")
> lines(filter(co2, rep(1/12, 12), side = 1), col = 2)
```



Lissage : Kernel

Un kernel (ou noyau) est une fonction de pondération qui permet de lisser les données (*cf.* fonction de densité d'une distribution). On peut donc l'appliquer à des séries temporelles.

Il existe plusieurs noyaux : uniforme, triangle, gaussien, daniell, etc...

Le noyau ne dépend que d'un paramètre qui va influer sur le niveau de lissage (m).

Lissage : Kernel

Deux étapes sont nécessaires.

On créé d'abord la fonction de pondération avec `kernel`. On essaye ici deux noyaux de Daniell avec un paramètre *m* différent :

```
> k1 <- kernel("daniell", 50)
> k2 <- kernel("daniell", 10)
```

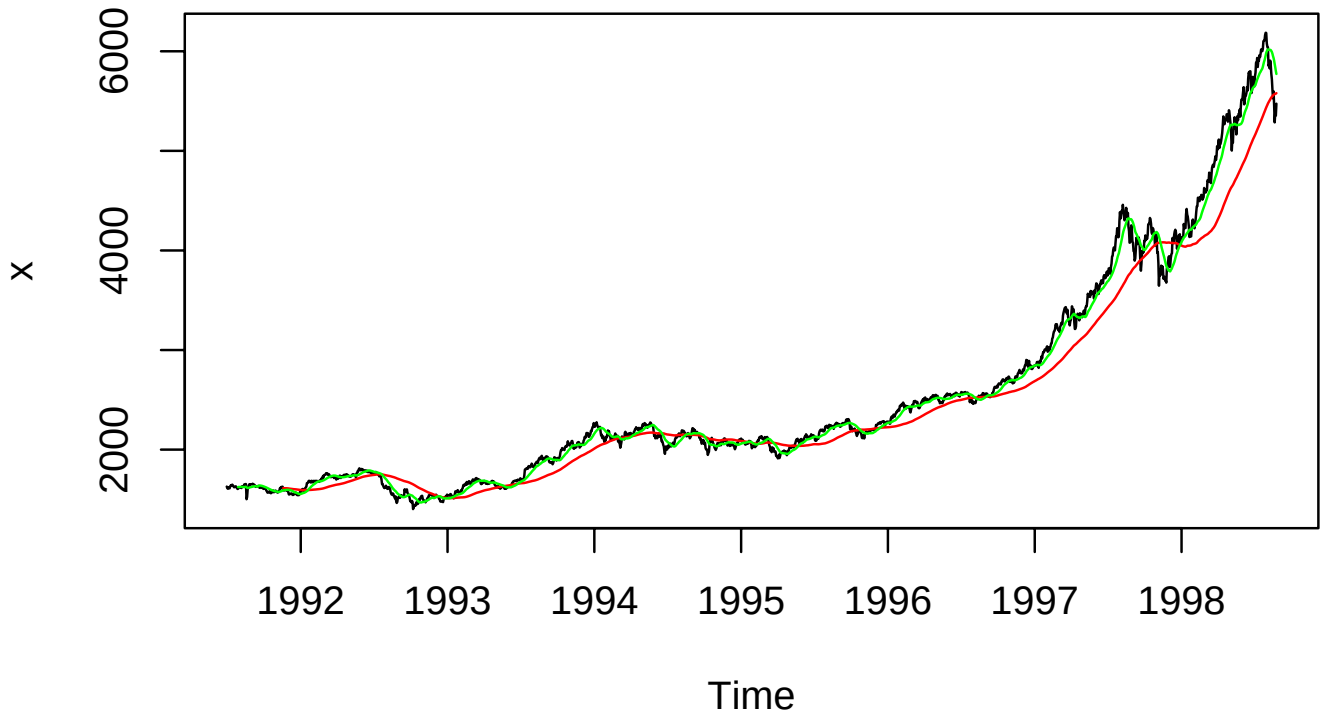
Puis on l'appplique aux données – ici des indices Euro Stock – avec la fonction `kernapply` :

```
> x <- EuStockMarkets[, 1]
> x1 <- kernapply(x, k1)
> x2 <- kernapply(x, k2)
```

Lissage : Kernel

On regarde ce que ça donne :

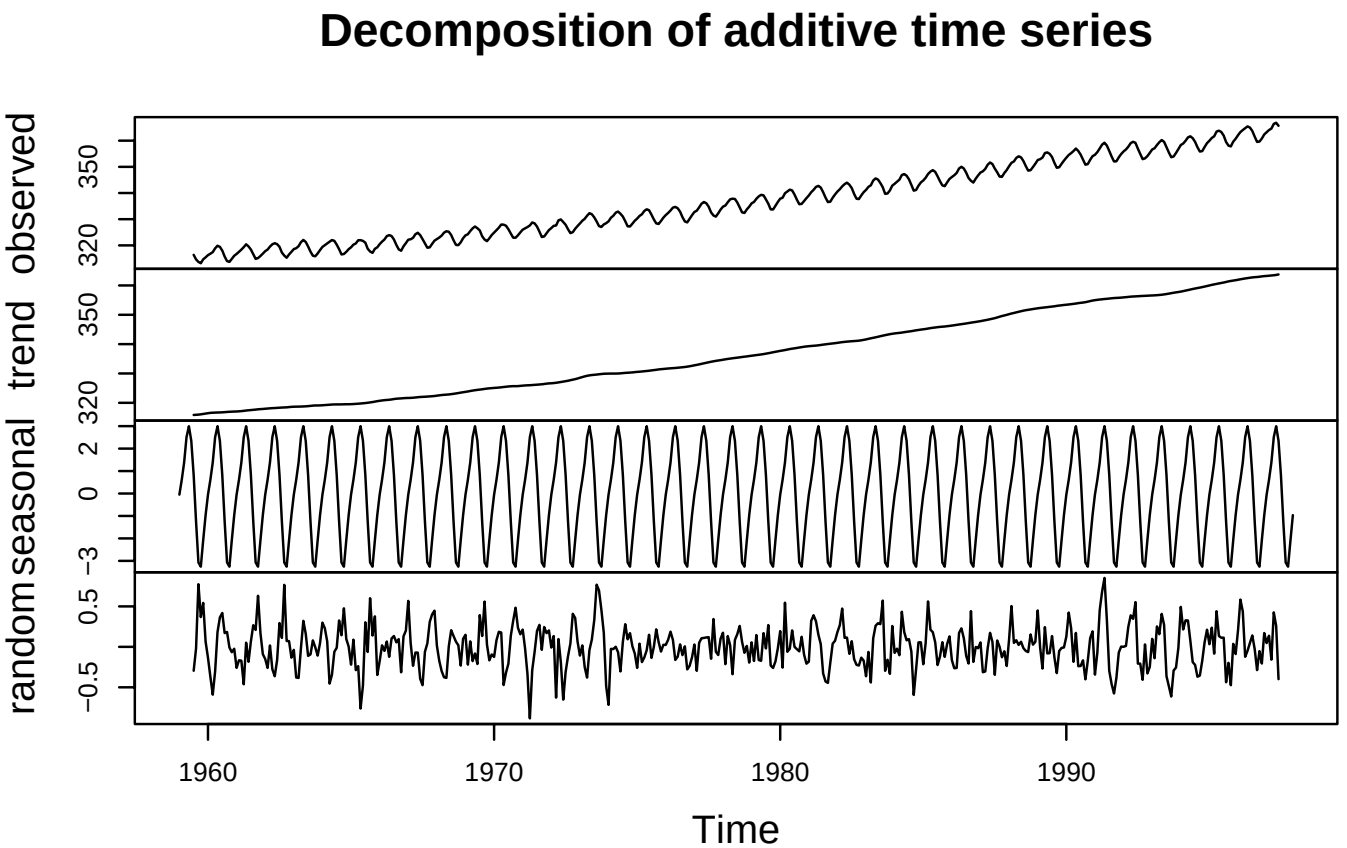
```
> plot(x)
> lines(x1, col = "red")
> lines(x2, col = "green")
```



Décomposition

Décomposition additive (ou multiplicative) en tendance $+(\times)$ variations saisonnières $+(\times)$ bruit :

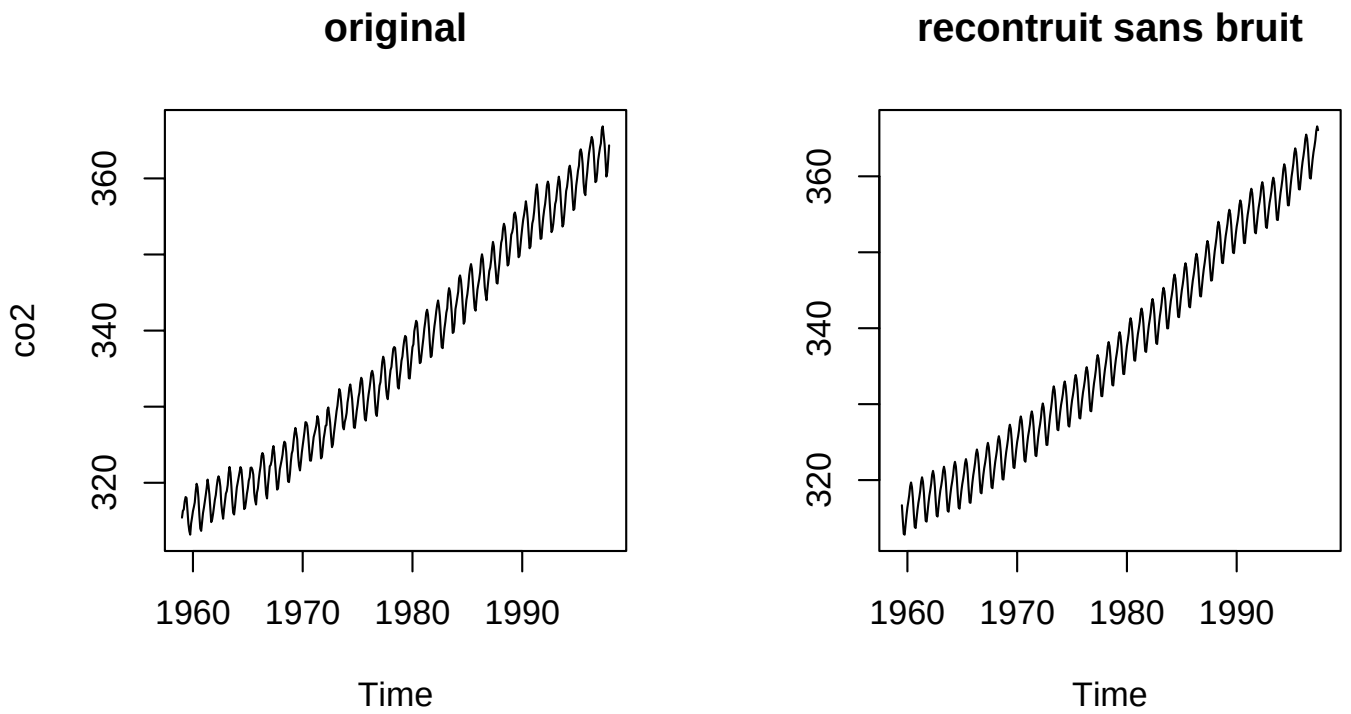
```
> r <- decompose(co2)
> plot(r)
```



Décomposition

On peut alors recomposer le signal sans le "bruit" :

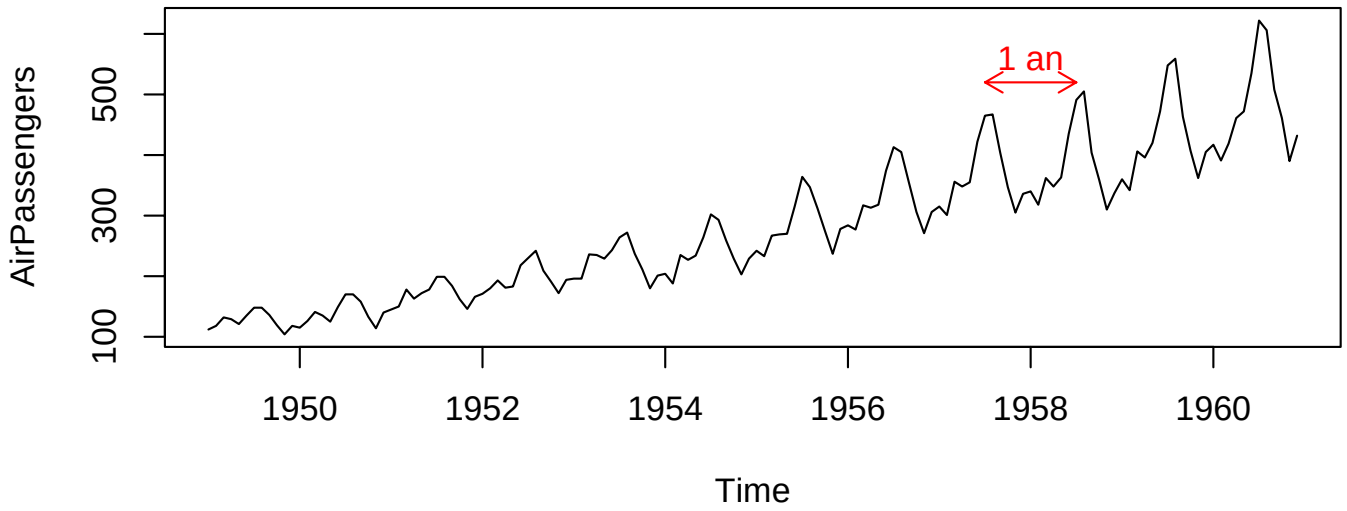
```
> recons <- r$trend + r$seasonal
> par(mfrow = c(1, 2))
> plot(co2, main = "original")
> plot(recons, main = "recontruit sans bruit", ylab = "")
```



Autocorrélation

On étudie les variations mensuelles du nombre de passagers d'une compagnie aérienne. On s'aperçoit qu'il y a des pics annuels :

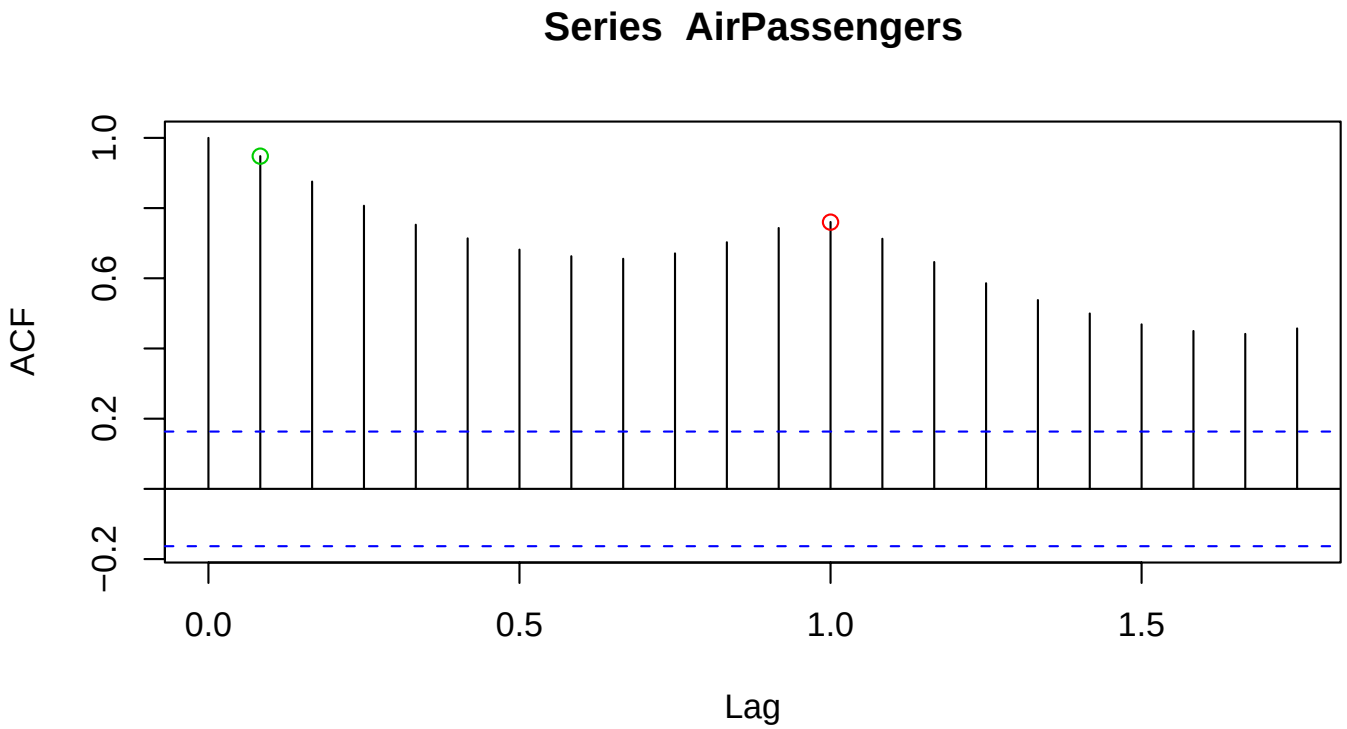
```
> data(AirPassengers)
> plot(AirPassengers)
> arrows(x0 = 1957.5, y0 = 520, x1 = 1958.5, y1 = 520, col = 2,
+     code = 3, length = 0.1)
> text(x = 1958, y = 560, "1 an", col = 2)
```



Autocorrélation

On retrouve cette périodicité (rouge) sur l'autocorrélation avec en plus un pic mensuel (vert) :

```
> acf(AirPassengers)
> points(x = 1, y = 0.76, col = 2)
> points(x = 1/12, y = 0.948, col = 3)
```



Références

- Livres
  - Cryer, JD & Chan, K-S (2008) – *Time series Analysis with applications in R*. Springer, 491 p.
  - Shumway, DH & Stoffer, DS (2006) – *Time Series Analysis and its applications with R Examples*. Springer, 575 p.
- Page web
  - page de Vincent Zoonekynde – http ://zoonek2.free.fr/
  - CRAN Tasks – http ://cran.at.r-project.org/web/views/TimeSeries.html