

# ANALYSE DES SÉRIES TEMPORELLES AVEC R



## 1 - Rappels sur l'environnement de travail de R

ALAIN QUARTIER-LA-TENTE  
Insee



---

**R** : Logiciel statistique

RStudio : IDE le plus connu

CRAN : Plateforme centralisant un ensemble de packages R sous un format normalisé permettant une installation facile

GitHub : Plateforme de partage de code où l'on retrouve de nombreux packages en développement

# Aide

---

- Si vous ne connaissez pas les fonctions, Google est votre ami
- Sinon `help(ma_fonction)` ou `?ma_fonction` pour chercher l'aide associée à `ma_fonction`. Voir aussi vignettes (documentation long format). Exemple :

```
# Pour voir l'ensemble des vignettes du package grid  
vignette(package = "grid")  
# Pour afficher une vignette en utilisant son nom  
vignette("moveline", package = "grid")
```

- Cran Task Views (<https://cran.r-project.org/web/views/>) regroupement de packages selon des thèmes particuliers. Exemple pour ce cours : <https://cran.r-project.org/web/views/TimeSeries.html>

# Sommaire

---

## 1. Les types de base

1.1 Les vecteurs

1.2 Les matrices

1.3 Les listes

1.4 Le data.frame et tibble

## 2. Importation des données

## 3. Les séries temporelles

# Les vecteurs (1)

---

Les vecteurs sont les objets les plus simples : créés avec fonction `c()` et leurs éléments peuvent être manipulés avec l'opérateur `[]`

```
v1 <- c(1, 2, 3); v2 <- c("a", "b")  
v1
```

```
## [1] 1 2 3
```

```
v2
```

```
## [1] "a" "b"
```

```
# v1 peut aussi se créer de façon équivalente avec :  
1:3
```

```
## [1] 1 2 3
```

```
# Pour concaténer deux vecteurs, notez le changement de type  
v3 <- c(v1, v2)  
v3
```

## Les vecteurs (2)

---

```
## [1] "1" "2" "3" "a" "b"
```

```
v3[c(4, 1)] # 4e puis 1er élément
```

```
## [1] "a" "1"
```

```
v3[-c(4, 1)] # on enlève 1er et 4e éléments
```

```
## [1] "2" "3" "b"
```

```
# Les éléments peuvent également être nommés
```

```
v4 <- c(elem1 = 1, elem2 = 2, 4)
```

```
v4
```

```
## elem1 elem2
```

```
##      1      2      4
```

```
names(v4)
```

```
## [1] "elem1" "elem2" ""
```

## Les vecteurs (3)

---

```
names(v4)[1] <- "toto"  
v4
```

```
## toto elem2  
##      1      2      4
```

```
v4[c("toto", "elem2")]
```

```
## toto elem2  
##      1      2
```

# Les matrices (1)

Matrices : vecteurs à deux dimensions créés avec fonction `matrix()`

```
m1 <- matrix(1:12, ncol = 3); m2 <- matrix(1:12, nrow = 3)
m1; t(m1); m1 * 2
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12

##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```



## Les matrices (2)

```
##      [,1] [,2] [,3]
## [1,]    2  10  18
## [2,]    4  12  20
## [3,]    6  14  22
## [4,]    8  16  24
```

```
m1 %*% m2 # multiplication matricielle
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   38   83  128  173
## [2,]   44   98  152  206
## [3,]   50  113  176  239
## [4,]   56  128  200  272
```

```
m1[, 1] # 1ere colonne : c'est un vecteur
```

```
## [1] 1 2 3 4
```

## Les matrices (3)

---

```
m1[-2, ] # Tout sauf 2ème ligne
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    3    7   11
## [3,]    4    8   12
```

```
# Nombre de lignes et de colonnes :
nrow(m1); ncol(m1); dim(m1)
```

```
## [1] 4
```

```
## [1] 3
```

```
## [1] 4 3
```

## Les matrices (4)

*# De la même façon que pour les vecteurs on peut nommer lignes/col*

```
colnames(m1) <- paste0("col", 1:ncol(m1))
```

```
rownames(m1) <- paste0("row", 1:nrow(m1))
```

```
m1
```

```
##      col1 col2 col3
```

```
## row1    1    5    9
```

```
## row2    2    6   10
```

```
## row3    3    7   11
```

```
## row4    4    8   12
```

```
m1[, "col2"]
```

```
## row1 row2 row3 row4
```

```
##     5    6    7    8
```

*# Pour combiner des matrices, on peut utiliser cbind et rbind:*

```
cbind(m1, 1:4)
```

## Les matrices (5)

---

```
##      col1 col2 col3
## row1    1    5    9 1
## row2    2    6   10 2
## row3    3    7   11 3
## row4    4    8   12 4
```

```
rbind(m1, m1)
```

```
##      col1 col2 col3
## row1    1    5    9
## row2    2    6   10
## row3    3    7   11
## row4    4    8   12
## row1    1    5    9
## row2    2    6   10
## row3    3    7   11
## row4    4    8   12
```

## Les matrices (6)

---

On peut utiliser la fonction `apply` pour appliquer une fonction à toutes les lignes ou toutes les colonnes. Exemple :

```
apply(m1, 1, sum) # somme sur toutes les lignes (dimension 1)
```

```
## row1 row2 row3 row4  
## 15 18 21 24
```

```
apply(m1, 2, sum) # somme sur toutes les colonnes (dimension 2)
```

```
## col1 col2 col3  
## 10 26 42
```

# Les listes (1)

---

Une liste peut contenir tout type d'objet

```
l1 <- list(v1, m1, v4); l1
```

```
## [[1]]  
## [1] 1 2 3  
##  
## [[2]]  
##      col1 col2 col3  
## row1    1    5    9  
## row2    2    6   10  
## row3    3    7   11  
## row4    4    8   12  
##  
## [[3]]  
##  toto elem2  
##    1     2     4
```

## Les listes (2)

---

```
length(l1) # nombre d'éléments d'une liste
```

```
## [1] 3
```

```
# On peut encore nommer les éléments de la liste :
```

```
names(l1) <- c("vect1", "mat", "vect2")
```

```
l1
```

```
## $vect1
```

```
## [1] 1 2 3
```

```
##
```

```
## $mat
```

```
##      col1 col2 col3
```

```
## row1    1    5    9
```

```
## row2    2    6   10
```

```
## row3    3    7   11
```

```
## row4    4    8   12
```

## Les listes (3)

---

```
##  
## $vect2  
## toto elem2  
##      1      2      4
```

```
# Pour accéder à un élément d'une liste utiliser [[,  
# autrement on a encore une liste  
l1[[1]] # liste d'un seul élément : v1
```

```
## $vect1  
## [1] 1 2 3
```

```
l1[[1]] # premier élément de la liste
```

```
## [1] 1 2 3
```

```
# On concatène deux listes avec fonction c:  
c(l1, l1[-2])
```



## Les listes (4)

---

```
## $vect1
## [1] 1 2 3
##
## $mat
##      col1 col2 col3
## row1   1   5   9
## row2   2   6  10
## row3   3   7  11
## row4   4   8  12
##
## $vect2
## toto elem2
##      1      2      4
##
## $vect1
## [1] 1 2 3
##
```

## Les listes (5)

---

```
## $vect2  
## toto elem2  
##      1      2      4
```

# Le data.frame (1)

---

Entre les listes et matrices : comme un tableur, souvent utilisé pour stocker des données

```
d1 <- data.frame(col1 = c("a", "b", "c"), col2 = 1:3)
d1
```

```
##   col1 col2
## 1    a    1
## 2    b    2
## 3    c    3
```

# Le tibble (1)

---

tibble : comme un data.frame réinventé, plus permissif

```
library(tibble)
t1 <- tibble(col1 = c("a", "b", "c"), col2 = 1:3)
t1 # ou as.tibble(d1)
```

```
## # A tibble: 3 x 2
##   col1    col2
##   <chr> <int>
## 1 a         1
## 2 b         2
## 3 c         3
```

## Le tibble (2)

---

```
# On peut aussi les définir ligne par ligne :  
tribble(  
  ~col1, ~col2,  
  "a", 1,  
  "b", 2,  
  "c", 3  
)
```

```
## # A tibble: 3 x 2  
##   col1  col2  
##   <chr> <dbl>  
## 1 a      1  
## 2 b      2  
## 3 c      3
```

# Sommaire

---

1. Les types de base
- 2. Importation des données**
3. Les séries temporelles

# Importer des données

---

Soyez fainéants et commencez par utiliser l'interface de RStudio (Environnement > Import Dataset).

# Sommaire

---

1. Les types de base
2. Importation des données
- 3. Les séries temporelles**



## ts()

---

Il existe de nombreux formats pour gérer les séries temporelles. Dans cette formation nous verrons :

- `ts()` : format de base R simple à utiliser mais des difficultés à gérer les fréquences non-entières (journalières, hebdomadaires, etc.)
- `tsibble()` : inspiré du tidyverse (tidyverts <https://tidyverts.org>) mais pour la gestion des séries temporelles

## ts() (1)

---

On peut créer un objet avec la fonction `ts(data = ., start = ., frequency = .)`

```
x = ts(c(1:12), start = 2020, frequency = 4)
x; class(x)
```

```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 2020     1     2     3     4
## 2021     5     6     7     8
## 2022     9    10    11    12

## [1] "ts"
```

```
mts <- ts(matrix(rnorm(30), 10, 3), start = c(1961, 1),
                frequency = 12)
mts; class(mts)
```

## ts() (2)

---

```
##           Series 1      Series 2      Series 3
## Jan 1961 -0.6264538  1.51178117  0.91897737
## Feb 1961  0.1836433  0.38984324  0.78213630
## Mar 1961 -0.8356286 -0.62124058  0.07456498
## Apr 1961  1.5952808 -2.21469989 -1.98935170
## May 1961  0.3295078  1.12493092  0.61982575
## Jun 1961 -0.8204684 -0.04493361 -0.05612874
## Jul 1961  0.4874291 -0.01619026 -0.15579551
## Aug 1961  0.7383247  0.94383621 -1.47075238
## Sep 1961  0.5757814  0.82122120 -0.47815006
## Oct 1961 -0.3053884  0.59390132  0.41794156

## [1] "mts"      "ts"       "matrix"
```

Pour manipulations : voir TP

# tsibble (1)

```
library(tsibble)
tsibbledata::aus_production
```

```
## # A tsibble: 218 x 7 [1Q]
##   Quarter Beer Tobacco Bricks Cement Electricity Gas
##   <qtr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1956 Q1 284 5225 189 465 3923 5
## 2 1956 Q2 213 5178 204 532 4436 6
## 3 1956 Q3 227 5297 208 561 4806 7
## 4 1956 Q4 308 5681 197 570 4418 6
## 5 1957 Q1 262 5577 187 529 4339 5
## 6 1957 Q2 228 5651 214 604 4811 7
## 7 1957 Q3 236 5317 227 603 5259 7
## 8 1957 Q4 320 6152 222 582 4735 6
## 9 1958 Q1 272 5758 199 554 4608 5
## 10 1958 Q2 233 5641 229 620 5196 7
```

## tsibble (2)

```
## # ... with 208 more rows
```

```
tsibbledata::global_economy
```

```
## # A tsibble: 15,150 x 9 [1Y]
```

```
## # Key:      Country [263]
```

##	Country	Code	Year	GDP	Growth	CPI	Imports	Ex
##	<fct>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
##	1 Afghanistan	AFG	1960	537777811.	NA	NA	7.02	
##	2 Afghanistan	AFG	1961	548888896.	NA	NA	8.10	
##	3 Afghanistan	AFG	1962	546666678.	NA	NA	9.35	
##	4 Afghanistan	AFG	1963	751111191.	NA	NA	16.9	
##	5 Afghanistan	AFG	1964	800000044.	NA	NA	18.1	
##	6 Afghanistan	AFG	1965	1006666638.	NA	NA	21.4	
##	7 Afghanistan	AFG	1966	1399999967.	NA	NA	18.6	
##	8 Afghanistan	AFG	1967	1673333418.	NA	NA	14.2	
##	9 Afghanistan	AFG	1968	1373333367.	NA	NA	15.2	

## tsibble (3)

---

```
## 10 Afghanistan AFG      1969 1408888922.      NA      NA      15.0
## # ... with 15,140 more rows
```

```
as_tsibble(mts)
```

```
## # A tsibble: 30 x 3 [1M]
## # Key:           key [3]
##       index key      value
##       <mtm> <chr>    <dbl>
## 1 1961 jan Series 1 -0.626
## 2 1961 fév Series 1  0.184
## 3 1961 mar Series 1 -0.836
## 4 1961 avr Series 1  1.60
## 5 1961 mai Series 1  0.330
## 6 1961 jui Series 1 -0.820
## 7 1961 jul Series 1  0.487
## 8 1961 aoû Series 1  0.738
```

## tsibble (4)

---

```
## 9 1961 sep Series 1 0.576
## 10 1961 oct Series 1 -0.305
## # ... with 20 more rows
```

S'adapte assez bien au tidyverse : `index_by()` remplace le `group_by()` mais sur les dates, `group_by_key()` permet de le faire sur les clefs :

```
library(dplyr)
as_tsibble(mts) %>%
  index_by() %>%
  summarise(moy = mean(value))
```

## tsibble (5)

---

```
## # A tsibble: 10 x 2 [1M]
##       index      moy
##       <month>   <dbl>
## 1 1961 jan    0.601
## 2 1961 fév   0.452
## 3 1961 mar  -0.461
## 4 1961 avr  -0.870
## 5 1961 mai   0.691
## 6 1961 jui  -0.307
## 7 1961 jul   0.105
## 8 1961 août  0.0705
## 9 1961 sep   0.306
## 10 1961 oct  0.235
```

```
as_tsibble(mts) %>%
  # index_by() %>%
  summarise(moy = mean(value))
```



## tsibble (6)

```
## # A tsibble: 10 x 2 [1M]
##       index      moy
##       <mtth>    <dbl>
## 1 1961 jan  0.601
## 2 1961 fév  0.452
## 3 1961 mar -0.461
## 4 1961 avr -0.870
## 5 1961 mai  0.691
## 6 1961 jui -0.307
## 7 1961 jul  0.105
## 8 1961 aoû  0.0705
## 9 1961 sep  0.306
## 10 1961 oct 0.235
```

```
as_tsibble(mts) %>%
  index_by(date = ~ yearquarter(.)) %>%
  summarise(moy = mean(value))
```

## tsibble (7)

---

```
## # A tsibble: 4 x 2 [1Q]
##   date      moy
##   <qtr> <dbl>
## 1 1961 Q1  0.198
## 2 1961 Q2 -0.162
## 3 1961 Q3  0.161
## 4 1961 Q4  0.235
```

```
as_tsibble(mts) %>%
  group_by_key() %>%
  summarise(moy = mean(value))
```

## tsibble (8)

---

```
## # A tsibble: 30 x 3 [1M]
## # Key:          key [3]
##   key          index    moy
##   <chr>        <mth>  <dbl>
## 1 Series 1 1961 jan -0.626
## 2 Series 1 1961 fév  0.184
## 3 Series 1 1961 mar -0.836
## 4 Series 1 1961 avr  1.60
## 5 Series 1 1961 mai  0.330
## 6 Series 1 1961 jui -0.820
## 7 Series 1 1961 jul  0.487
## 8 Series 1 1961 aoû  0.738
## 9 Series 1 1961 sep  0.576
## 10 Series 1 1961 oct -0.305
## # ... with 20 more rows
```