

4 - Manipulation de Séries Temporelles en R

Séries Temporelles avec R - Initiation

Anna Smyk, Tanguy Barthelemy

Insee - Département des Méthodes Statistiques



Objectifs

Spécificité des séries temporelles: couple (index temps, valeur observée)

La date est liée aux valeurs, contenue dans l'objet

Fonctions spécifiques adaptées à cette structure : extractions, jointures, graphiques, autocorrelations, interpolations, lissage, modélisation, décomposition...

Besoin: utiliser des fonctions pré-codées dans des packages R

(éviter de recoder)

Nombreuses fonctions et packages disponibles:

voir CRAN Task View: <https://cran.r-project.org/web/views/TimeSeries.html>

Objectifs

Selon les besoins statistiques: différents packages requièrent différents formats

Deux exemples:

- rjdverse (famille autour de JDemetra+): objets de classe TS (très courant)

voir rjdverse: <https://github.com/rjdverse>

- fpp3 (forecasting principles and practice): objets de classe tsibble (prolonge la grammaire du tidyverse, permet de garder d'autres variables que la date et la valeur)

voir autour de fpp3: <https://robjhyndman.com/software/>

De multiples standards...

- objets ts : package stats
- objets tsibble : package tsibble: <https://CRAN.R-project.org/package=tsibble>
- objets zoo package zoo: <https://CRAN.R-project.org/package=zoo>
- objets xts package xts: <https://CRAN.R-project.org/package=xts>

...et un convertisseur

Convertisseur : package tsbox <https://CRAN.R-project.org/package=tsbox>

- conversion d'un format à l'autre
- nombreuses fonctions agnostiques

cf: cheat sheet

Manipulation de dates:

- package auxiliaire lubridate: <https://lubridate.tidyverse.org/>

cf. cheat sheet

Principales opérations présentées dans cette séquence

créations d'objets de classe TS (univariés et multivariés) et tsibble

- conversions from and to data frames

Manipulations de données

- extractions de sous-séries
- extractions d'attributs
- jointures et création de séquences de dates

Fonctions statistiques

- sommes, moyennes
- imputation de valeurs manquantes

On se concentre sur les objets de classe TS (dans une moindre mesure tsibble)

Création d'objets de classe TS univariés I

Fonction `ts(data = ., start = ., frequency = .)`

- à partir d'un vecteur numérique (colonne de data frame...)

Définition avec longueur, date de début et fréquence

```
ts1 <- ts((1:24) + 100, start = c(2020, 1), frequency = 12)  
print(ts1)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2020	101	102	103	104	105	106	107	108	109	110	111	112
2021	113	114	115	116	117	118	119	120	121	122	123	124

```
class(ts1)
```

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

Création d'objets de classe TS univariés II

- frequency est le nombre d'observations par unité de temps (ici année) : 1=annuelle, 2=semestrielle, 4=trimestrielle, 6=bi-mestrielle, 12=mensuelle

Définition avec longueur, date de fin et fréquence

```
ts(3 + 5 * rnorm(60), frequency = 4, end = c(2002, 2)) #dernier point inclus
```

	Qtr1	Qtr2	Qtr3	Qtr4
1987			9.46456199	-0.40635980
1988	1.12575210	7.38118539	3.20324206	3.23750275
1989	-7.23177204	8.37748495	8.35455712	9.86289939
1990	0.32950626	3.01605580	22.13423984	-7.49897763
1991	1.97233695	1.47501051	2.92123192	-6.40903124
1992	6.94411912	-0.51523309	-2.81372479	-1.90291738
1993	3.10220953	8.11948281	3.44017193	9.09544461
1994	-0.87949280	-1.11160270	-3.53979237	9.25357376
1995	-7.52734846	0.67109372	-0.07796041	3.90992416
1996	2.77477325	13.33920129	2.83431241	-1.10221111

Création d'objets de classe TS univariés III

```
1997 -2.91100059  6.23439372  3.48460198  6.48152968
1998  3.32607156 -0.63370852  2.44632599 -1.36146269
1999 12.02569192  2.61284044 11.65146856  3.76967390
2000 -3.20143948  9.94238167  4.33374306  3.11469937
2001  3.47385637  3.10720723  4.49125406  8.38936736
2002  6.94457792 -6.19579092
```

Définition avec date de début et de fin

```
ts(3 + 5 * rnorm(72), frequency = 12, start = c(2000, 1), end = c(2004, 12)) #coupe le vecteur
```

	Jan	Feb	Mar	Apr	May	Jun
2000	0.83301932	-2.60644602	1.43135223	5.94174814	16.43106347	-1.63218455
2001	7.52141534	10.73694703	6.38384093	2.77578727	1.82827410	9.00607885
2002	-2.84805502	6.63723721	3.64172770	2.22470637	9.07073487	7.77937910
2003	1.60313272	2.41643264	5.29098137	-2.62956439	0.04479241	4.26277516
2004	0.89784776	2.42238027	4.65672590	8.27995043	2.94207436	2.81388228
	Jul	Aug	Sep	Oct	Nov	Dec

Création d'objets de classe TS univariés IV

2000	-1.08698343	-7.13938321	7.10310744	2.44972811	10.42098979	13.49551868
2001	-3.01294183	-4.00019670	2.72487714	13.41333868	6.84508118	2.81802009
2002	-2.03828147	8.98578817	3.86081128	-2.73445150	1.13228644	-9.59201108
2003	3.06615800	-0.97398014	6.40043102	-7.65974375	3.64190953	5.74046021
2004	2.88807566	-7.77243590	8.52067974	-3.47333097	8.28746857	5.48087389

Création d'objets de classe TS multivariés I

A partir d'une matrice

```
mts_object <- ts(  
  matrix(rnorm(30), 12, 3),  
  start = c(2000, 1),  
  frequency = 12  
)  
print(mts_object)
```

	Series 1	Series 2	Series 3
Jan 2000	-0.20021665	1.8184464	0.06300698
Feb 2000	-0.67463673	-0.2878569	0.97676323
Mar 2000	0.26242087	0.4438804	-0.58200109
Apr 2000	1.23347646	0.9804488	-0.03288991
May 2000	-0.29137814	-0.4524981	-0.37808921
Jun 2000	0.39031095	-2.4755831	-0.84651349
Jul 2000	-0.74794814	-0.8210066	-0.20021665
Aug 2000	-0.04926114	0.7893263	-0.67463673

Création d'objets de classe TS multivariés II

```
Sep 2000  1.45589776  0.0254428  0.26242087
Oct 2000  0.22713953  0.3954131  1.23347646
Nov 2000 -1.40053668  0.1366299 -0.29137814
Dec 2000 -0.22580884 -1.3833592  0.39031095
```

```
class(mts_object)
```

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

```
is.mts(mts_object)
```

```
[1] TRUE
```

Création d'objets de classe TS multivariés I

A partir d'un data frame: on extrait les colonnes numériques (matrice de valeurs) et on respécifie les dates lors de la création de l'objet mts (attention à la date de début)

```
# data frame ipi
y_raw <- ts(ipi[, "RF3030"], start = c(1990, 1), frequency = 12)
y_raw

# start = c(1990,1): résulte de la connaissance du data frame
```

Récupération d'attributs (1/2) I

```
ts1 <- ts((1:24) + 100, start = c(2020, 1), frequency = 12)  
start(ts1)
```

```
[1] 2020    1
```

```
class(start(ts1))
```

```
[1] "numeric"
```

```
start(ts1)[2]
```

```
[1] 1
```

```
end(ts1)
```

```
[1] 2021    12
```

```
frequency(ts1)
```

```
[1] 12
```

Récupération d'attributs (2/2) I

création de la série des dates correspondante à un objet ts : fonction `time()`

```
time(ts1) #fractions: 1/frequency  
  
# fonctions pour retrouver un format date  
# exemple  
date <- zoo::as.Date(time(ts1))  
date  
class(date)
```

Récupération de la position dans l'année d'une observation : fonction `cycle()`

```
cycle(ts1)  
class(cycle(ts1))
```

Extraction et jointures I

Exemple avec deux objets ts

```
ts1 <- ts(1:15, start = c(2022, 1), frequency = 12)
ts2 <- ts(13:24, start = c(2023, 1), frequency = 12)
```

- extraction `ts.window` ou `tsbox::ts_span`

```
ts11 <- window(ts1, start = c(2022, 6), end = c(2022, 12))
ts11
```

```
      Jun Jul Aug Sep Oct Nov Dec
2022   6   7   8   9  10  11  12
```

```
ts12 <- ts_span(ts1, "-6 month")
ts12
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2022                10  11  12
2023  13  14  15
```


Extraction et jointures II

Union

```
# séries en tableau  
# on garde toute la couverture temporelle en rajoutant des NA  
  
ts.union(ts1, ts2) #classe mts  
  
#head(ts.union(ts1,ts2))
```

Intersection

```
# on ne garde que les périodes communes  
ts.intersect(ts1, ts2)
```

Extraction et jointures III

Conversions avec le package `tsbox`

`ts_c`: comme `ts.union`

`ts_bind`: on combine plusieurs séries en une, si chevauchement la première citée l'emporte (sauf si NA), cf. exemples infra

`ts_chain`: comme `ts_bind` mais avec interpolation

Listes de séries I

Format liste pratique pour appliquer des fonctions avec la famille 'lapply()'

```
ma_liste <- ts_tslist(mts_object)
ma_liste[2]
```

\$`Series 2`

	Jan	Feb	Mar	Apr	May	Jun
2000	1.8184464	-0.2878569	0.4438804	0.9804488	-0.4524981	-2.4755831
	Jul	Aug	Sep	Oct	Nov	Dec
2000	-0.8210066	0.7893263	0.0254428	0.3954131	0.1366299	-1.3833592

```
class(ma_liste[2])
```

[1] "list"

```
ma_liste[[2]]
```

Listes de séries II

	Jan	Feb	Mar	Apr	May	Jun
2000	1.8184464	-0.2878569	0.4438804	0.9804488	-0.4524981	-2.4755831
	Jul	Aug	Sep	Oct	Nov	Dec
2000	-0.8210066	0.7893263	0.0254428	0.3954131	0.1366299	-1.3833592

```
class(ma_liste[[2]])
```

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

Opérations arithmétiques sur les séries I

```
ts1 <- ts(1:6, start = c(2023, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2023                                1   2
2024    3   4   5   6
```

```
ts2 <- ts(10:15, start = c(2024, 1), frequency = 12)
ts2
```

```
      Jan Feb Mar Apr May Jun
2024   10  11  12  13  14  15
```

```
# opérations simples: sur périodes communes (coupe)
ts1 + ts2 # idem pour - * /
```

```
      Jan Feb Mar Apr
2024   13  15  17  19
```

Opérations arithmétiques sur les séries II

```
# avec ts box
```

```
# périodes communes
```

```
ts1 %ts+% ts2
```

```
      Jan Feb Mar Apr  
2024  13  15  17  19
```

```
# on peut forcer le format de la série figurant à gauche
```

```
ts_df(ts1) %ts+% ts2
```

```
      time value  
1 2024-01-01    13  
2 2024-02-01    15  
3 2024-03-01    17  
4 2024-04-01    19
```

Manipulation de dates I

création de séquences de dates sous R avec la fonction `seq()`

```
date <- seq(from = as.Date("2024-01-01"),  
            to = as.Date("2024-12-31"),  
            by = "month")
```

date

```
[1] "2024-01-01" "2024-02-01" "2024-03-01" "2024-04-01" "2024-05-01"  
[6] "2024-06-01" "2024-07-01" "2024-08-01" "2024-09-01" "2024-10-01"  
[11] "2024-11-01" "2024-12-01"
```

```
date <- seq(from = as.Date("2024-01-01"),  
            to = as.Date("2024-12-31"),  
            by = "quarter")
```

date

```
[1] "2024-01-01" "2024-04-01" "2024-07-01" "2024-10-01"
```

Manipulation de dates I

Manipulation avec le package lubridate (voir cheat sheet) qui contient de très nombreuses fonctions, ici deux exemples:

- conversion au format date d'une chaîne de caractères, fonctions `ymd()`, `ymd_hms`, `dmy()`, `dmy_hms`, `mdy()`

```
"Jan-2020"
```

```
[1] "Jan-2020"
```

```
"Jan-2020" %>% class()
```

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

```
date <- lubridate::my("Jan-2020")  
date
```

```
[1] "2020-01-01"
```


Manipulation de dates II

```
class(date)
```

```
[1] "Date"
```

Manipulation de dates I

- extraction d'attributs/modification de la composante d'une date avec les fonctions `year()`, `month()`, `mday()`, `hour()`, `minute()` and `second()`

```
# création d'une variable date
date <- seq(from = as.Date("2024-01-01"),
           to = as.Date("2024-03-31"),
           by = "month")
date
```

```
[1] "2024-01-01" "2024-02-01" "2024-03-01"
```

```
month(date)
```

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

```
month(date) %>% class()
```

```
[1] "numeric"
```

Manipulation de dates II

```
month(date[2]) <- 11  
date
```

```
[1] "2024-01-01" "2024-11-01" "2024-03-01"
```

Série retardée I

Pour calculer la série retardée/avancée, il suffit d'utiliser la fonction `lag()`, mais attention au paramétrage selon le package

```
ts1 <- ts(1:6, start = c(2024, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun
2024   1   2   3   4   5   6
```

```
# package stats
stats::lag(ts1, k = -1) # attention période série finale
```

```
      Feb Mar Apr May Jun Jul
2024   1   2   3   4   5   6
```

```
# package dplyr sur vecteur numérique
dplyr::lag(as.vector(ts1), 1)
```

```
[1] NA  1  2  3  4  5
```

Série retardée II

```
# package tsbox  
tsbox::ts_lag(ts1) #k=1 par défaut
```

	Feb	Mar	Apr	May	Jun	Jul
2024	1	2	3	4	5	6

```
tsbox::ts_lag(ts1, 12)
```

	Jan	Feb	Mar	Apr	May	Jun
2025	1	2	3	4	5	6

```
tsbox::ts_lag(ts1, 4)
```

	May	Jun	Jul	Aug	Sep	Oct
2024	1	2	3	4	5	6

Différenciation I

Différenciation - à l'ordre k

$$Diff(k) = X_t - X_{t-k}$$

- Le plus souvent à l'ordre 1 (tendance) et/ou à l'ordre 12,4... saisonnalité

$$Diff(1) = X_t - X_{t-1}$$

$$Diff(12) = X_t - X_{t-12}$$

```
ts1 <- ts(1:24, start = c(2024, 1), frequency = 12)
ts1
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	1	2	3	4	5	6	7	8	9	10	11	12
2025	13	14	15	16	17	18	19	20	21	22	23	24

```
# diff d'ordre 1
diff1 <- ts1 - lag(as.vector(ts1))
diff1
```

Différenciation II

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	NA	1	1	1	1	1	1	1	1	1	1	1
2025	1	1	1	1	1	1	1	1	1	1	1	1

```
diff1 <- ts1 - ts_lag(ts1) #attention NA et période de la série finale  
diff1
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024		1	1	1	1	1	1	1	1	1	1	1
2025	1	1	1	1	1	1	1	1	1	1	1	1

```
# ou fonction directe  
ts_diff(ts1)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	NA	1	1	1	1	1	1	1	1	1	1	1
2025	1	1	1	1	1	1	1	1	1	1	1	1

Différenciation III

```
# diff d'ordre 12
diff12 <- ts1 - ts_lag(ts1, 12)
diff12
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2025	12	12	12	12	12	12	12	12	12	12	12	12

```
# ou fonction directe
ts_diffy(ts1)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2025	12	12	12	12	12	12	12	12	12	12	12	12

Agrégation I

Passer à une fréquence plus basse avec une fonction spécifique (somme, moyenne, dernière valeur)

exemple de solution : `tsbox::ts_frequency`

```
ts1 <- ts((1:12) + 100, start = c(2024, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2024  101 102 103 104 105 106 107 108 109 110 111 112
```

```
ts_frequency(ts1, "quarter") #default: mean
```

```
      Qtr1 Qtr2 Qtr3 Qtr4
2024   102   105   108   111
```

```
ts_frequency(ts1, "quarter", "sum")
```

```
      Qtr1 Qtr2 Qtr3 Qtr4
2024   306   315   324   333
```

Agrégation II

```
ts_frequency(ts1, "quarter", "last")
```

	Qtr1	Qtr2	Qtr3	Qtr4
2024	103	106	109	112

Désagrégation temporelle vers une fréquence plus élevée : problème plus complexe, voir packages `rjd3bench`,...

Summary statistics I

Fonctions usuelles : moyenne, médiane, écart-type

```
ts1 <- ts((1:24) + 100, start = c(2024, 1), frequency = 12)  
ts1
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	101	102	103	104	105	106	107	108	109	110	111	112
2025	113	114	115	116	117	118	119	120	121	122	123	124

```
mean(ts1)
```

```
[1] 112.5
```

```
median(ts1)
```

```
[1] 112.5
```

```
sd(ts1)
```

```
[1] 7.071068
```

Summary statistics I

Par type de période : solution possible fonctions du package xts : `apply.monthly()`, `apply.quarterly()`, `apply.yearly()`..

Pour cela il faut auparavant convertir les données au format xts.

Par exemple pour calculer la moyenne annuelle :

```
library(xts)
#| echo: true
#| eval: true
ts1 <- ts((1:24) + 100, start = c(2023, 1), frequency = 12)
ts1
moy_an <- xts::apply.yearly(as.xts(ts1), mean)
moy_an
```

Séries temporelles avec données auxiliaires I

Le format `tsibble` permet de garder des variables descriptives dans une même table de données avec les séries temporelles

- Les séries temporelles sont empilées selon une clé (key)
- la date est stockée dans `index`

Ce format:

- permet d'utiliser le package `fable` et outils connexes
- offre de nombreuses possibilités graphiques
- se manipule bien avec la grammaire `tidyverse` (`dplyr`)

Voir <https://tsibble.tidyverts.org/>

Exemple de tsibble

```
library("tsibble")  
mts_object  
as_tsibble(mts_object)
```

voir exemples de données dans le package tsibbledata <https://tsibbledata.tidyverts.org/>

Valeurs manquantes I

On peut utiliser des fonctions du package zoo ou imputeTS (par exemple) pour

- repérer les valeurs manquantes : fonction `is.na`
- Les enlever: au début et/ou à la fin `zoo::na.trim()`
- Les imputer
 - dernière valeur `zoo::na.locf`
 - interpolation linéaire `zoo::na.approx()`
 - autres méthodes: moyenne, splines, kalman filter

Voir package imputeTS (cheat sheet)

Valeurs manquantes I

```
ts1 <- ts((1:12) + 100, start = c(2024, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2024 101 102 103 104 105 106 107 108 109 110 111 112
```

```
#ajout NA début
```

```
ts2 <- ts(as.numeric(rep(NA, 2)), start = c(2023, 12), frequency = 12)
ts2
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2023                                     NA
2024  NA
```

```
ts12 <- ts_bind(ts1, ts2)
ts12
```


Valeurs manquantes II

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2023												NA
2024	101	102	103	104	105	106	107	108	109	110	111	112

```
# ajout de NA au milieu  
month(as.Date(time(ts12))) # pas de NA ici
```

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

```
ts12[month(as.Date(time(ts12))) %in% c(3, 8)] <- NA  
ts12
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2023												NA
2024	101	102	NA	104	105	106	107	NA	109	110	111	112

```
#on enlève les valeurs manquantes du début  
ts12_i <- zoo::na.trim(ts12, sides = "left")  
ts12_i
```

Valeurs manquantes III

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	101	102	NA	104	105	106	107	NA	109	110	111	112

```
ts12_ii <- imputeTS::na_mean(ts12_i) # moyenne de la série sans NA  
ts12_ii
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	101.0	102.0	106.7	104.0	105.0	106.0	107.0	106.7	109.0	110.0	111.0	112.0