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/Timeséries.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                            7.60803763  1.49247800
1988  2.10622546 -0.43299820 -8.63021771  9.35784931
1989 -4.77614560  3.46547080  0.66995047 -3.41010689
1990  5.68544313  0.88249298 -5.54756809 -2.30301464
1991 -0.66185664  2.16924061 -5.23519112 -7.51376218
1992 12.30326999  8.50965908 10.29382675  2.78007921
1993  3.67831356 -8.54986682 -0.46930954  0.58755293
1994  2.01868838  6.58000037 13.09821772 -0.81509974
1995  8.20926614  4.65661733  4.29270971  5.12346163
1996 14.84450261  3.62638028 -0.84935946  5.56997444
```

Création d'objets de classe TS univariés III

```
1997  0.24828706  4.17592088  4.36600076 11.63936600
1998 -4.60628120 11.94579235  3.86181871  5.72119326
1999 -3.57763232  4.41275443  3.44085324  4.31971565
2000  0.05059824 -1.91494876  6.65903970  6.84971693
2001  2.81839066 12.01912421  5.16751975 -0.18709009
2002 -4.95492279  2.79728644
```

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   8.5617517   2.7860182  16.1024103  -2.4561594   3.0862941  -2.9525919
2001  -4.0913684   1.7339267  -0.7783138  -0.6437029   2.9644852   4.7282366
2002   5.0716514   4.7650414   2.2968687  12.8362415   2.8919448   2.8713392
2003  -5.0628196   6.1123743   5.4594903   9.9249650   2.9500081  -2.7306009
2004  -0.2996163   0.2624452   0.8247454  -2.1605511  11.9997857   9.3147345
             Jul         Aug         Sep         Oct         Nov         Dec
```

Création d'objets de classe TS univariés IV

```
2000  -1.6709147   1.8672423  14.2225768  -1.5403196   2.2284121  14.2732689
2001   9.9514256  -3.6297407  -4.3562100   0.1122368   3.9179531  -1.6519327
2002   6.6890092 -10.5330466  -4.6699607   7.8123799   5.1901218  -6.0271659
2003   8.0959164   6.7417064   5.6254228  -3.0841393   3.8313383  -1.6722854
2004   8.7754432   3.0395946   6.1836422  11.0685984   4.9791673   3.3130484
```

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 -0.1843435 -1.49911974  0.2460619
Feb 2000 -1.6306795  0.72367587 -0.6236767
Mar 2000  2.7057728  0.74768473 -1.0895719
Apr 2000  1.0580748 -0.04976409 -0.2588394
May 2000 -0.5722323 -0.02643647 -0.5964659
Jun 2000  0.2622455  1.22461142  1.1982204
Jul 2000  1.0760201  0.54990775 -0.1843435
Aug 2000  0.8838356  0.17546890 -1.6306795
```

Création d'objets de classe TS multivariés II

```
Sep 2000 -0.6844504  0.17244406  2.7057728
Oct 2000  0.1419599 -0.49548051  1.0580748
Nov 2000 -0.5284532 -0.76714014 -0.5722323
Dec 2000  1.6675998 -1.82004796  0.2622455
```

```
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

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

```
[1] 2020    1
```

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

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

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

```
[1] 1
```

```r
end(ts1)
```

```
[1] 2021   12
```

```r
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 -1.49911974  0.72367587  0.74768473 -0.04976409 -0.02643647  1.22461142
             Jul         Aug         Sep         Oct         Nov         Dec
2000  0.54990775  0.17546890  0.17244406 -0.49548051 -0.76714014 -1.82004796
```

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

```
[1] "list"
```

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

Listes de séries II

```
             Jan         Feb         Mar         Apr         May         Jun
2000 -1.49911974  0.72367587  0.74768473 -0.04976409 -0.02643647  1.22461142
             Jul         Aug         Sep         Oct         Nov         Dec
2000  0.54990775  0.17546890  0.17244406 -0.49548051 -0.76714014 -1.82004796
```

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

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

Opérations arithmétiques sur les séries I

```r
ts1 <- ts(1:6, start = c(2023, 11), frequency = 12)
ts1
```

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

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

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

```r
# 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

```r
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 parmetrage 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 defaut
```

```
     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
```

```r
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
```

```r
# 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

```r
# 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
```

```r
# 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 élevée 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

```r
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 (apr 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
```

```r
# 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
```

```r
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
```

```r
#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
```

```r
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
```