2 - Manipulation de Séries Temporelles en R

Webinaire: désaisonnalisation avec JDemetra+ en R

Anna Smyk

Objectifs (1/2)

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 (2/2)

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

Objets TS et principales opérations

On se concentre sur les objets TS utiles pour utiliser JDemetra+ !

création d'objets de classe TS (univariés et multivariés)

- 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

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                                7.697864843  2.012990583
1988  1.255609508  2.848082763  0.592764342  5.414040004
1989  8.523225516  3.371181196 14.276838036  4.176312505
1990  1.397267255  1.146080470  0.625202757  0.791531484
1991 -4.507362510  6.555723904  1.759749024  9.730578288
1992  6.615997119 -0.009239469  8.624077287 10.713358162
1993  8.459180061 -5.231411661 16.332953621 -1.799153700
1994  2.303302884  5.721454627  8.555093631  5.964840284
1995  0.675053420 -2.795691031 -2.408284846  6.297188661
1996  5.697395685 -2.216796705  7.339205822 -1.827856552
```

Création d'objets de classe TS univariés III

```
1997  7.364232233  3.085877601 -4.002738636 -7.443916215
1998  6.229823140  6.967077902 -0.302362443 -2.046222000
1999  7.439066161  1.820575630  1.121979811  8.909510832
2000  1.506706071  1.591303591 -0.495414679  3.332618673
2001 11.902351059  3.912202901  0.950996021 -6.640472888
2002  4.517002476  9.174323491
```

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  2.3879519  2.2426528  9.1443423 -0.8699061  5.6553036  9.9318930
2001 -9.8984668  6.3093401  2.5188711 -3.6496905  7.7383733 -1.0218702
2002 -1.4349603  5.9986621 -1.6131607 -3.7672393 -5.5152211 -7.5920123
2003 -5.1062256  4.5231781 14.9629514  0.5351813  0.4246297  9.1834535
2004  7.0896040  9.5889582  8.6343500 -0.5811934  1.3870590  3.0123453
            Jul        Aug        Sep        Oct        Nov        Dec
```

Création d'objets de classe TS univariés IV

```
2000  5.2129306  2.5368502  8.3273051  8.2408655  3.5549664  4.2811989
2001 -1.2934309  4.1203109  9.7568833 14.0568559 -2.4277761  5.5958500
2002  2.3307042  7.7165011  0.6732602  6.6869846  1.8194692  2.9696732
2003 -0.2358497  8.7431116  5.0415068  5.4250506  8.4969065  2.0798751
2004  1.1807495  6.7554062  6.4045560 -0.6595265  7.8463247 -1.2893808
```

Création d'objets de classe TS multivariés I

A partir d'une matrice

```r
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 -0.6826716 -1.558477705  0.1068675
Feb 2000 -0.1276200 -1.448388151 -0.2646749
Mar 2000  0.7703443 -0.370159866  1.3701741
Apr 2000 -2.0830983  0.592546901  0.8018274
May 2000 -1.2274683  0.005438883 -1.4662555
Jun 2000 -1.4257901  1.488788875 -0.4958323
Jul 2000  1.7079209 -0.395394979 -0.6826716
Aug 2000  0.5714481 -0.020089872 -0.1276200
```

Création d'objets de classe TS multivariés II

```
Sep 2000  1.4244197  0.304726721  0.7703443
Oct 2000  1.5346440  0.791423810 -2.0830983
Nov 2000 -0.2318498  0.057347986 -1.2274683
Dec 2000  1.0201726  0.529137522 -1.4257901
```

```
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
2000 -1.558477705 -1.448388151 -0.370159866  0.592546901  0.005438883
              Jun           Jul           Aug           Sep           Oct
2000  1.488788875 -0.395394979 -0.020089872  0.304726721  0.791423810
              Nov           Dec
2000  0.057347986  0.529137522
```

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

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

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

Listes de séries II

```
              Jan           Feb           Mar           Apr           May
2000 -1.558477705 -1.448388151 -0.370159866  0.592546901  0.005438883
              Jun           Jul           Aug           Sep           Oct
2000  1.488788875 -0.395394979 -0.020089872  0.304726721  0.791423810
              Nov           Dec
2000  0.057347986  0.529137522
```

```
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 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

```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,..

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

```r
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
```

```r
#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
```

```r
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
```